Contribution au domaine de la conception des Systèmes Embarqués et Pervasifs Faible Consommation

Johann Laurent

HABILITATION A DIRIGER LES RECHERCHES

UNIVERSITE DE BRETAGNE-SUD

sous le sceau de l'Université européenne de Bretagne

présentée par

Johann Laurent

Préparée au Lab-STICC (UMR 6285)

Laboratoire des Sciences et Techniques de l'Information de la communication et de la Connaissance

Contribution au domaine de la conception des Systèmes Embarqués et Pervasifs Faible Consommation

HDR soutenue le 05 novembre 2015
devant le jury composé de :

Christian Piguet
Head of SoC Program, CSEM, Neuchatel
Professeur à l'Ecole Polytechnique de Lausanne – Rapporteur

Emmanuel Casseau
Professeur à l'Ecole Nationale Supérieure des Sciences Appliquées et de Technologie – Rapporteur

Michel Robert
Professeur à L'université de Montpellier 2 – Rapporteur

Eric Martin
Professeur à l'Université de Bretagne Sud – Examinateur

Jean Philippe Diguet
Directeur de recherche CNRS au Lab-STICC – Examinateur

Jean Philippe Babau
Professeur à l'Université de Bretagne Occidentale - Examinateur

Avant-Propos

Je tiens tout d'abord à remercier messieurs Christian Piguet, Emmanuel Casseau et Michel Robert qui me font l'honneur d'être rapporteur de ce mémoire d'habilitation à diriger les recherches. Je remercie également messieurs Eric Martin, Jean-Philippe Diguet et Jean-Philippe Babau d'avoir accepté de prendre part au jury en tant qu'examinateurs.

Les travaux présentés dans ce mémoire sont le fruit de nombreuses collaborations avec différents membres du Lab-STICC que je tiens à remercier car cela m'a permis d'aborder plusieurs thématiques au cours de ces dernières années. Je profite aussi de ce mémoire pour remercier tous les doctorants passés et actuels (Antoine, Saadia, Ronan, Jean Christophe, Mathilde, Hugo et Erwan) avec lesquels j'ai travaillé et sans qui je n'écrirais pas ce mémoire aujourd'hui.

Depuis mes débuts à l'Université de Bretagne Sud, j'ai la chance de côtoyer énormément de personnes sympathiques au sein de l'UFR SSI. Je voudrais remercier aussi tous les membres du laboratoire qui contribuent à faire du Lab-STICC un lieu où il fait bon vivre et travailler, à commencer par Guy Gogniat son directeur. Une pensée particulière pour Florence et Virginie pour tout ce qu'elles font pour nous toute l'année avec le sourire et la bonne humeur qui les caractérisent.

Enfin merci à toutes les personnes qui me supporte ou m'ont supporté au cours de ces dernières années et je sais que ce n'est pas tous les jours facile.

Johann Laurent Mai 2015

Table des matières

Chapitre 1

Introduction générale

Introduction

Ce manuscrit présente une synthèse de mes travaux de recherche, d'enseignements et administratifs réalisés depuis février 2005, date de ma nomination en tant que Maître de conférences à l'Université de Bretagne Sud. Mes travaux de recherche ont été effectués au sein du laboratoire LESTER jusqu'en 2008 puis au laboratoire Lab-STICC au sein de l'équipe Méthodes et Outils pour les Circuits et Systèmes Numériques (MOCS) jusqu'à ce jour. J'ai choisi de présenter mes contributions de recherche par thématique et non de façon chronologique afin d'obtenir une structuration du manuscrit par grands thèmes abordés.

L'introduction générale est divisée en quatre sections. La première est un curriculum vitae, la seconde présente une vue d'ensemble de mes différentes activités de recherche, la troisième dresse ma liste de publications post doctorales. La dernière partie de cette introduction présente mes diverses activités d'enseignement. Cette introduction générale est suivie de 4 chapitres dont 3 qui abordent de manière plus détaillée mes contributions scientifiques.

Le chapitre 2 est consacré à l'Estimation de la Consommation des Architectures Logicel. Ce travail est en continuité de mes travaux de thèse et ont démarré avec le projet SPICES piloté par Eric Senn. Ce projet avait pour but, pour notre partie, de modéliser et d'estimer la consommation des services d'un système d'exploitation à haut niveau. Ces travaux ont fait l'objet de la thèse de Saadia Dhouib (2006-2009) co-dirigée par Eric Senn et Jean Philippe Diguet.

Le chapitre 3 présente les travaux autour de l'estimation et l'optimisation de la consommation des interconnexions dans les systèmes sur puce (SoC). Dans un système sur puce la consommation d'énergie générée par les interconnexions peut devenir non négligeable ; il devient donc indispensable de pouvoir optimiser cette consommation. Afin de pouvoir juger des optimisations proposées, un modèle d'estimation est nécessaire car le temps de conception et de simulation (au niveau électrique) est prohibitif. Ces travaux ont fait l'objet de la thèse d'Antoine Courtay (2005-2008) co dirigée par Olivier Sentieys et Nathalie Julien.

Le chapitre 4 aborde mes derniers travaux de recherche sur la conception de systèmes pervasifs pour le domaine maritime. Ces travaux abordent plusieurs sous thèmes comme:

- la mesure de la performance pour la course au large ; travaux de thèse de Ronan Douguet (2010-2014)
- l'utilisation de la réalité augmentée pour l'aide à la navigation ; travaux de thèse de Jean Christophe Morgère (2011-2015)
- l'optimisation temps réel d'énergies renouvelables pour voilier du futur ; travaux de thèse de Mathilde Tréhin (2013- ?)
- les algorithmes et plateforme faible consommation pour la conception d'un pilote automatique haute performance pour le nautisme ; travaux de thèse d'Hugo Kerhascoet (2014-2017).

Seule la partie des travaux sur la mesure de performance sera développée dans ce document.

Le chapitre 5 tire les conclusions des différents travaux présentés dans ce document et expose quelques perspectives associées.

Une annexe, à la suite, regroupe une publication représentative de chacun des thèmes abordés dans ce document.

- Un article publié dans la revue Journal Of Low Power Electronics, co-écrit avec Saadia Dhouib, Eric Senn, Jean Philippe Diguet et Dominique Blouin, traite de l'estimation de la consommation d'énergie et de puissance au niveau système dans le cadre de systèmes embarqués implantant un système d'exploitation. Cet article met en exergue les travaux présentés dans le chapitre 2.

- Un article publié dans la revue Journal Of Low Power Electronics, co-écrit avec André Rossi et Marc Sevaux, se focalise sur l'optimisation du placement des données en mémoire pour les architectures logiciel. Ces travaux se sont poursuivis dans la thèse de Maria Soto ; thèse co dirigée par André Rossi et Marc Sevaux au sein du Lab-STICC.

- Un article publié dans la revue Journal Of Low Power Electronics, co-écrit avec Antoine Courtay, Olivier Sentieys et Nathalie Julien, aborde l'estimation et l'optimisation de la consommation des interconnexions dans les systèmes sur puce. Cet article met en lumière les travaux présentés dans le chapitre 3.

- Un article publié dans la revue Journal of Sailboat Technology, co-écrit avec Ronan Douguet, Jean Philippe Diguet et Yann Riou, traite de l'estimation du leeway dans la mesure de performance dans le cadre de la course au large. Cet article présente un des aspects des travaux présentés dans le chapitre 4.

1.1 Curriculum Vitae

1.1.1 Données personnelles

Nom : Laurent

Prénom : Johann

Date de Naissance : 17 mai 1974 à Vannes (Morbihan)

Nationalité : Française

Adresse professionnelle :

Lab-STICC - UMR CNRS 6285

Centre de recherche

2 rue Saint Maudé

BP922116

56321 Lorient Cedex

Téléphone : +033 2 97 87 45 63

Email : johann.laurent@univ-ubs.fr

Web : http://www-labsticc.univ-ubs.fr/~laurent

Adresse personnelle:

28 rue de lanmeur

29360 Clohars Carnoët

1.1.2 Expérience professionnelle

Depuis 2005 Maître de Conférences à l'Université de Bretagne Sud, Lorient (Lab-STICC & UFR Sciences et Sciences de l'Ingénieur)

2003-2004 ATER temps complet à l'Université de Bretagne Sud

2002-2003 ATER à mi-temps à l'Université de Bretagne Sud

1999-2002 Doctorant au Laboratoire d'Electronique et des Systèmes Temps Réels

1.1.3 Formation

2002 Doctorat à l'Université de Bretagne Sud, Estimation de la consommation dans la conception système des applications embarquées temps réels

1999 DEA d'électronique à SUPELEC Rennes

1998 Maîtrise EEA à l'Université de Bretagne Sud

1.1.4 Encadrement de thèses soutenues

- Co-encadrement de Jean Christophe Morgère avec Jean Philippe Diguet, Mobile Augmented Reality System for Marine Navigation (2011-2015). M. Jean Christophe Morgère est actuellement ingénieur de recherche pour la SATT Bretagne Valorisation sur le projet RAM qui fait suite à ses travaux de thèse.
- Co-encadrement de Ronan Douguet avec Jean Philippe Diguet, Optimisation de la mesure et de l'interprétation des performances dans le cadre de la course au large (2010-2014). M. Ronan Douguet est actuellement ingénieur au sein de la société MARPORT.
- Co-encadrement de Saadia Dhouib avec Jean Philippe Diguet et Eric Senn, Système Level Power Estimation (2006-2009). Mlle Saadia Dhouib est actuellement ingénieur de recherche au CEA LIST.
- Co-encadrement d'Antoine Courtay avec Olivier Sentieys et Nathalie Julien, Consommation d'énergie dans les interconnexions sur puce : Estimation de haut niveau et optimisations architecturales (2005-2008). M. Antoine Courtay est aujourd'hui Maître de Conférences à l'ENSATT Lannion.

1.1.5 Encadrement de thèse en cours

- Co-encadrement de Mathilde Tréhin avec Jean Philippe Diguet, l'optimisation temps réel d'énergies renouvelables pour voilier du futur (20013- ?)
- Co-encadrement d'Hugo Kerhascoët avec Eric Senn, Algorithmes et plateforme faible consommation pour le développement d'un pilote automatique haute performance (2014-2017)
- Co-encadrement d'Erwan Moréac avec André Rossi, Optimisation de la latence et de la consommation d'énergie dans les réseaux sur puce (2014-2017)

1.1.6 Participation à des jurys de thèse

- Dr Jean Christophe Morgère (UBS, 2015)
- Dr Ronan Douguet (UBS, 2014)
- Dr Saadia Dhouib (UBS, 2009)

- Dr Antoine Courtay (UBS, 2008)

1.1.7 Autres activités relevant de l'encadrement doctoral

- Encadrement d'étudiants de master recherche : Maria Mendez (2013), Jonas Desfontaine(2014), Emmanuel Riberolles (2014), Thomas Toublanc (2014), Guillaume Glon(2014).
- Encadrement de stages de Master 2 : Florent Rouxel (2014), Adrien Quignon (2012), Moline Yoann (2011).
- Encadrement de stage de Master 1 : Maria Mendez (2013)

1.1.8 Activités scientifiques nationales

- Membre du GDR SoC/SiP
- Membre du GDR ISIS thème C
- Orateur invité par le groupe Eurolarge afin de promouvoir la conception de systèmes embarqués contraints dans la filière du nautisme.
- Orateur invité aux séminaires du Lab-STICC afin de présenter les recherches sur les systèmes pervasifs.

1.1.9 Projets :

- Projet MOBIFLEX en 2005-2008 avec l'équipe CAIRN de l'IRISA financé par la région Bretagne.
- Projet ITEA SPICES (Support for Predictable Integration of mission Critical Embedded Systems) en 2006-2009 avec entre autre les sociétés AIRBUS, Axlog, Thales Avionics, Thalès Communication, BARCO… Le projet a été financé par le ministère de l'Industrie pour les partenaires français.
- Projet RAM (Réalité Augmentée Mobile) en 2011-2014 financé par la région Bretagne et le CG56.
- Projet BoWI (Body World Interaction) en 2012-2015 avec l'équipe CAIRN de l'IRISA, Télécom Bretagne, l'IETR, l'INSA de Rennes financé par le Labex CominLabs.
- Projet GreenVideo en 2013-2016 avec les sociétés Thalès Communications & Security, Ektacom, Vitec, Thomson Video Network, TeamCast, l'IETR. Ce projet est financé sur les Fonds Uniques Interministériels.
- Projet AMI Voilier du Futur en 2013- ? avec les sociétés NKE, Multiplast, Grand Large Yachting, CRAIN, VPLP. Ce projet est financé via l'ADEME par les fonds d'investissements d'avenir.
- Projet SmartMast en 2014 avec la société Lorima financé en propre par la société.

1.1.10 Implication dans la communauté scientifique

- Editorial Board Member de la revue American Journal of Computer Science and Information Engineering édité par l'American Association for Sciences and Tehcnology
- Membre du comité de programme de la conférence DASIP 2010-2012
- Membre du comité d'organisation de l'Ecole thématique Conception Faible Consommation pour les systèmes embarqués et temps réels 2014

- Relecteur pour les revues internationales :
    - Springer Journal of Real Time Image Processing,
    - ACM Transaction on Embedded Computing Systems,
    - ACM Transaction on Design Automation of Electronics Systems,
    - EURASIP Journal of Embedded Systems,
    - IEEE Transaction on Computer Aided of Integrated Circuits and Systems.
    - IEEE Transaction On Very Large Scale Integration
- Relecteur pour les conférences internationales:
    - IEEE Design Automation Conference : 2010-2011-2014
    - IEEE International Symposium on Mixed and Augmented Reality : 2014
    - IEEE Design and Test in Europe: 2013
    - IEEE Great Lakes Symposium on Very Large Scale Integration: 2009-2011-2013
- Expert STIC auprès du pôle de compétitivité Pôle Mer Bretagne depuis 2011
- Expert STIC auprès d'EUROLARGE depuis 2012
- Co pilote de l'axe Information Communication Technologies pour l'Océan (ICTO) du Lab-STICC
- Participation aux comités de suivi de thèse :
    - Quang-Hai Khuat (IRISA-CAIRN)
    - Ganda Stéphane (IRISA-CAIRN)
- Animateur à la fête de la science à l'UBS depuis 2005
- Animateur à la coupe de robotique RobotFesta depuis 2008

1.1.11 Responsabilités administratives

- Directeur adjoint du département Sciences et Techniques de l'UFR SSI de l'UBS (2009-2012)
- Membre de la commission Recherche et Personnel de l'UFR SSI (2011-2013)
- Membre du comité de pilotage du Master STIC (depuis 2007)
- Membre du comité de pilotage de la licence Sciences et Techniques (2007-2011)
- Membre du comité de rédaction du Master STIC (2011)
- Directeur de la spécialité GEII du Master STIC (depuis 2008)
- Directeur d'étude du Master 2 GEII (depuis 2006)
- Directeur d'étude du Master 1 GEII (2008-2012)
- Membre élu du conseil du laboratoire CNRS Lab-STICC (depuis 2012)

1.2 Activités de recherche

Mon parcours de recherche a débuté par une thèse sur l'estimation de consommation d'architecture logiciel au sein du laboratoire LESTER de l'Université de Bretagne Sud. Suite à mon recrutement en tant que Maitre Conférences en 2005 au sein de l'UBS, j'ai eu la chance de co encadrer ma première thèse seulement après quelques mois ; cette première thèse s'est inscrite dans une collaboration avec l'équipe CAIRN de l'IRISA basée à Lannion. Celle-ci s'intéressait aux aspects optimisation et estimation de la consommation des réseaux d'interconnexions dans les systèmes sur puce. Cette section présentera succinctement les différentes recherches effectuées à la suite de cette première

thèse, leurs buts, les personnes impliquées ainsi que les publications afférentes. L'ensemble des publications est ensuite listé dans la section 1.3 et réparti en différentes catégories :

- Brevet (B)
- Ouvrage (O)
- Chapitre d'ouvrage (CO)
- Revue internationale (RI)
- Revue nationale (RN)
- Conférence internationale (CI)
- Conférence nationale (CN)

1.2.1 Estimation de la consommation des architectures logiciel

Equipe : Nathalie Julien, Eric Martin, Eric Senn – Université de Bretagne Sud ; Mohamed Abid, Jalel Ktari – Ecole Nationale de Sfax (Tunisie)

Publications : [RI01, RI02, RI03, RI04, RI05, RN01, CI01, CI02, CI03, CI04, CI05, CI06, CI07, CI08, CI09, CI10, CI11, CN01, CN02, CN03, CN04].

Ce thème de recherche porte presque exclusivement sur mes travaux de thèse et partiellement sur ceux de Jalel Ktari, étudiant tunisien, en co tutelle entre l'Université de Sfax et l'Université de Bretagne Sud. L'objectif de ses travaux était de permettre la réalisation d'estimation de la consommation d'une application exécutée sur une cible logiciel. La première étape de ces travaux a été la proposition d'une nouvelle méthode permettant la réalisation d'un modèle de consommation pour des processeurs, principalement des processeurs de traitement du signal (DSP). Cette méthode, appelée FLPA pour Functional Level Power Analysis, permettait de modéliser en consommation un processeur non plus par la modélisation de chacune des instructions composant son jeu mais en modélisant des blocs fonctionnels de son architecture. Cette méthode prennait en compte plusieurs phénomènes à la fois (consommation de la lecture de l'instruction, de son exécution, des ruptures de pipeline, des défauts de cache etc.) mais surtout diminuait fortement le temps d'obtention du modèle de consommation. Le modèle de consommation ainsi réalisé dépendait d'un certain nombre de paramètre qui pouvaient être extraits à partir du code de l'application. Dans un premier temps, l'extraction de ces paramètres était réalisée à partir du code assembleur généré après compilation ou directement écrit par l'utilisateur. Ensuite, nous avons proposé une méthode permettant d'extraire directement ces paramètres à partir du code écrit en langage C. Un outil d'estimation (SoftExplorer), basé sur ces 2 méthodes, a été développé par mes soins et est accessible librement sur demande ; une centaine d'équipe dans le monde utilise aujourd'hui l'outil. La méthode FLPA a été par la suite réutilisée avec succès pour modéliser la consommation d'autres systèmes (GPP, FPGA etc.). Nous l'avons également utilisée afin de modéliser les services d'un système d'exploitation, thématique présenté ci-dessous. Dans ce document, les recherches de ma thèse ne seront pas présentées plus avant.

1.2.2 Modélisation en consommation des services d'un système d'exploitation

Equipe : Jean Philippe Diguet – CNRS ; Dominique Blouin, Saadia Dhouib, Mickaël Lanoë, Eric Senn – Université de Bretagne Sud

Publications : [CO01, RI07, CI20, CI18, CI16, CI15, CI13].

Ces recherches sont dans la continuité de mes travaux de thèse en s'intéressant non plus à l'application exécutée par une architecture processeur mais à la consommation d'énergie engendrée par l'utilisation de services d'un système d'exploitation. Cette étude a été menée dans le cadre d'un projet Européen ITEA SPICES et a donné lieu entre autre aux travaux de thèse de Saadia Dhouib. Le projet se faisait dans le cadre d'applications aéronautiques critiques qui doivent être certifiées avant de pouvoir être embarquées dans un avion. Cette certification oblige, entre autre, de pouvoir simuler fonctionnellement le système dans son ensemble afin de vérifier que les diverses propriétés attendues sont réalisées. Le cycle de développement de tels produits implique la nécessité de réaliser des simulations dès les premières phases de développement. L'ensemble du système est, pour se faire, décrit dans le langage AADL (Architecture Analysis and Design Language) qui permet de décrire aussi bien la plateforme matériel que logiciel. La première étape de ces travaux était de fournir un modèle de consommation de l'OS prenant en compte l'architecture matériel utilisée (SoftCore sur FPGA, GPP, DSP). Cette modélisation s'est effectuée en se basant sur la méthodologie FLPA développée durant ma thèse. Ceci nous a permis de proposer un modèle paramétrique multi niveaux. Le deuxième apport de ces travaux a été la mise à disposition, pour la communauté AADL, d'un outil d'estimation de la consommation CAT (Consumption Analysis Tool) capable, à partir d'une description conjointe de la plateforme et de l'application, d'estimer la consommation de l'application et du système d'exploitation associé. Ces travaux seront présentés plus avant dans le chapitre 2 de ce document.

1.2.3 Optimisation du placement des données en mémoire

Equipe : Johann Laurent, André Rossi, Marc Sevaux, Maria Soto – Université de Bretagne Sud

Publications : [O01, RI9].

Ces travaux ont pour point de départ un constat réalisé lors de mes travaux de thèse. Lors de la compilation sur des architectures non munies de mémoires caches et de système d'exploitation, le placement des données en mémoire est très mal réalisé. En effet le compilateur exploite mal voire pas du tout l'architecture mémoire de la cible (possibilité d'accès parallèles) ce qui a pour conséquence d'engendrer des conflits d'accès à des bancs mémoire alors qu'ils sont évitables de par la conception de l'architecture. Ce problème apparait clairement comme un problème de recherche opérationnel qui vise à minimiser le nombre de conflits mémoire afin d'optimiser les performances temporelles et de consommation d'un système mémoire. Nous avons donc proposé une modélisation mathématique de ce problème et également une méthode de résolution s'appuyant sur une programmation linéaire en nombres entiers nous garantissant l'obtention d'une solution optimale. Cette méthode a été outillée et implantée dans l'outil MemExplorer. Ces travaux seront présentés plus avant dans l'annexe B.

1.2.4 Estimation de la consommation des interconnexions dans les SoCs

Equipe : Antoine Courtay, Nathalie Julien, Johann Laurent, Erwan Moréac, André Rossi – Université de Bretagne Sud ; Olivier Sentieys – Université de Rennes 1 / Equipe CAIRN IRISA

Publications : [B01, RI08, RI06, CI19, CI17, CI14, CI12, CN05, CN06].

Ces travaux ont été les premiers que j'ai co-encadrés à la suite de mon recrutement en tant que MCF et ont fait l'objet des travaux de thèse d'Antoine Courtay. L'idée de ces travaux était dans un premier

temps de modéliser la consommation énergétique des données transitant sur un lien de communication (interconnexion) implanté dans un système sur puce (SoC). En effet, le nombre de liens de communication explosant dans les systèmes actuels le coût énergétique de ceux-ci devient non négligeable dans le bilan énergétique global de la puce. Les interconnexions pouvant être parallèles sur une longueur non négligeable le phénomène de couplage capacitif (appelé Crosstalk) entre fil apparaît. Ce couplage a comme premier effet de modifier le délai de transition des informations sur le lien mais modifie également l'énergie nécessaire au transport de cette information. Nous avons donc proposé un tableau de coût de transitions énergétiques entre un fil victime et ses voisins afin de compléter le modèle physique des interconnexions. Nous avons démontré que, à l'inverse de ce qui était communément admis, le tableau des transitions énergétiques était différent de celui du facteur de délai. Comme toutes les optimisations de la consommation des interconnexions se basaient sur cette hypothèse, nous avons donc proposé une nouvelle approche d'optimisation, appelée Spatial Switching (breveté), permettant de supprimer les transitions les plus pénalisantes d'un point de vue énergétique. Ces travaux seront développés dans le chapitre 3 de ce document.

1.2.5 Optimisation de la consommation d'énergie et de la latence dans les réseaux sur puce (NoC)

Equipe : Johann Laurent, Erwan Moréac, André Rossi – Université de Bretagne Sud

Publications : [].

Ces travaux sur les réseaux sur puce viennent de démarrer suite à l'obtention d'une bourse CDE par Erwan Moréac (2014). L'idée de ces travaux est de proposer une ou plusieurs méthodes d'optimisation permettant à la fois l'amélioration de la latence et de l'énergie d'un réseau sur puce. En effet, la consommation de ces réseaux sur les puces actuelles devient non négligeable et doit faire l'objet d'une attention particulière. Dans un premier temps, nous nous focaliserons sur l'estimation de la consommation et de la latence. Aujourd'hui les modèles existants ignorent un phénomène important appelé Crosstalk (diaphonie capacitive) qui influe aussi bien sur le temps de propagation de l'information que sur la consommation engendrée par cette propagation. Nous allons donc proposer un nouveau modèle pour la partie lien de communication du réseau et l'intégrer dans le modèle classiquement utilisé (DSENT). Ce modèle complet du réseau sera ensuite intégré dans un simulateur de réseau sur puce existant que nous modifierons afin de le rendre cycle près/bit près (CABA). Ces travaux se font en continuité de ceux réalisés durant la thèse d'Antoine Courtay et en reprendront certains aspects. Ces travaux seront évoqués dans le chapitre 4 de ce document.

1.2.6 Optimisation de la mesure et de l'interprétation des performances dans le cadre de la course au large

Equipe : Ronan Douguet, Jean Philippe Diguet, Johann Laurent – Université de Bretagne Sud ; Yann Riou Groupama Sailing Team

Publications : [CI24, CI22, CI21].

Ces travaux s'inscrivent dans le cadre d'un financement Cifre avec le Groupama Sailing Team de Franck Cammas et ont fait l'objet des travaux de thèse de Ronan Douguet. La problématique majeure dans le domaine de la course sur des supports munis de voiles est la mesure de la force et de la direction du vent réel (vent naturel et non perçu sur le support). Bien qu'a priori il soit simple de

mesurer ce vent, le problème s'avère beaucoup plus complexe et nécessite la mise en place d'algorithmes permettant de corriger l'ensemble des perturbations de mesure du vent. De plus, le domaine applicatif (i.e. la mer) entraine des contraintes dans la conception du système embarqué qui doit être économe en énergie, peu encombrant et s'interfacer avec de nombreux protocoles de communication. Nous avons proposé une approche permettant de prendre en compte, en plus des perturbations habituelles, la dérive du bateau par rapport à l'eau lié au vent (leeway). En effet, la dérive globale d'un bateau est la somme de la dérive due au courant marin et due au vent (glissement du bateau par rapport à l'eau). Nous avons également développé une centrale ouverte permettant d'interfacer les différents capteurs présents sur un bateau de compétition mais également de calculer la direction et la force du vent réel. Ce système a été comparé avec une centrale de navigation du commerce classiquement utilisée par les équipes de course et est aujourd'hui intégrée sur tous les supports du Groupama Sailing Team. Ces travaux seront développés dans le chapitre 4 de ce document.

1.2.6 Réalité augmentée pour les applications marines

Equipe : Jean Philippe Diguet, Johann Laurent, Jean Christophe Morgère – Université de Bretagne Sud

Publications : [CI26, CI25, CI24].

Ces travaux ont fait l'objet d'un financement par la région Bretagne dans le cadre de la bourse ARED de Jean Christophe Morgère. Le point de départ de ces travaux est issu d'un constat qui est que lorsque nous sommes sur un bateau il est parfois difficile de se positionner et de s'avoir exactement où nous sommes. En effet, de nombreux accidents surviennent car les personnes à bord prennent de mauvaises décisions (mauvaises manœuvre, mauvais cap etc..). L'idée est donc de fournir un système embarqué capable d'afficher en surimpression à la réalité les informations importantes. Plusieurs problèmes sont donc à résoudre à savoir la localisation et l'orientation en temps réel, le tri d'information, la génération d'objets ainsi que la faible consommation afin de limiter la taille et le poids du système. La solution proposée durant ces travaux font aujourd'hui l'objet d'un projet de maturation avec la SATT Ouest Valorisation d'une durée de 12 mois. Ces travaux ne seront pas développés dans ce document.

1.2.7 Optimisation temps réel d'énergies renouvelables pour voiliers du futur

Equipe : Jean Philippe Diguet, Johann Laurent, Eric Senn, Mathilde Tréhin – Université de Bretagne Sud

Publications : [].

Ces travaux ont débuté dans le cadre du projet AMI Voilier du Futur et font l'objet de la thèse de Mathilde Tréhin. Le but de ce projet est de développer un voilier dont l'empreinte écologique est la plus faible possible en utilisant un maximum de matériaux bio-sourcés et en générant l'énergie électrique à bord à partir de ressources renouvelables (hydrogénérateur, éolien, photovoltaïque). Ce voilier étant fait pour le grand voyage, les contraintes énergétiques seront fortes car de nombreux éléments de confort et d'aide à la navigation y seront intégrés. L'idée de ces travaux est donc de fournir un système de gestion autonome permettant d'analyser en temps réels la consommation et la production d'énergie à bord et de proposer en fonction de ces données soit une dégradation de la

qualité de service de certains appareils, soit de proposer de suspendre certains services voire de proposer un nouveau routage du bateau afin de conserver l'ensemble des services. Un simulateur complet du voilier intégrant les différents modèles de consommation/production d'énergie a été développé. Il permet à partir d'une trace GPS, représentant le parcours du voilier, de simuler le parcours du bateau en fonction de la météo et de prévoir la production d'énergie et la consommation à bord en fonction de profil utilisateur (chaque profil correspond à des besoins en énergie différents). Afin de simuler au plus près le comportement du bateau sur la trace GPS, nous avons implanté une loi de commande qui simule les actions sur la barre du bateau en fonction de la météo (utilisation de GRIB météo réels) et des polaires du bateau (vitesse du bateau en fonction de l'angle et la force du vent). Suite à des problèmes de financements du projet, ces travaux sont aujourd'hui suspendus ; ils ne seront donc pas abordés dans ce document.

1.2.8 Algorithmes et plateforme faible consommation pour le développement d'un pilote automatique haute performance pour le nautisme

Equipe : Paul Fraisse, Hugo Kerhascoët, Pascal Merien – NKE Marine Electronics, Johann Laurent, Eric Senn – Université de Bretagne Sud, Frédéric Hauville – IRENAV

Publications : [CI en cours de soumission].

Ces travaux font l'objet d'un financement Cifre par la société NKE et sont réalisés par Hugo Kerhascoët. L'idée de ces travaux est d'améliorer la mesure du vent afin d'optimiser les lois de commande d'un pilote automatique. En effet, le pilote automatique d'un voilier doit réaliser des manœuvres afin de garder un cap ou un angle par rapport au vent. Même lorsque le pilote a pour consigne un cap à suivre, il utilise des informations de vent, de gîte etc… afin de calculer au mieux la commande à générer. Pour mesurer le vent, le capteur le plus couramment utilisé est l'anémo-girouette à coupelles situé en haut du mât. Cette position engendre un grand nombre de perturbations (mouvements du bateau, bras de levier lié au mât, torsion du mât, cisaillement du vent …) qui entraînent des erreurs de mesure qui vont induire des prises de décisions erronées de la part du pilote. Nous proposons donc d'intégrer de nouveaux capteurs aux capteurs de vent afin de s'affranchir de certaines perturbations, de fusionner des données issues de plusieurs capteurs et de prendre en compte le comportement mécanique des capteurs de vent. Tout ceci permettra à terme d'obtenir un système de mesure adaptatif temps réel pouvant être intégré dans une solution de pilote automatique. Bien entendu, l'aspect consommation d'énergie est important et devra être pris en compte lors du développement du système embarqué qui sera intégré dans la gamme de produits de l'entreprise NKE. Ces travaux ne seront pas plus avant développés dans ce document.

1.3 Liste de publications

1.3.1 Brevets (2)

B02 : «Système de surveillance», Johann Laurent, Jean Philippe Diguet, Yvan Eustache, FR 10 60744, 2010

B01: «Spatial Switching», Johann Laurent, Antoine Courtay, Olivier Sentieys, Nathalie Julien, FR 08 51672, 2008

1.3.2 Ouvrage (1)

O01 : «Memory Allocation Problems in Embedded Systems: Optimization Methods », Maria Soto, André Rossi, Marc Sevaux, Johann Laurent, Computer Engineering Series, ISTE, WILEY, ISBN 978-1-84821-428-6, 2013

1.3.3 Chapitre d'ouvrage (1)

CO01: «Power and energy consumption estimations in model based design », Eric Senn, Jean-Philippe Diguet, Johann Laurent, Saadia Douhib, Skander Turki, Dominique Blouin, Languages for Embedded Systems and their Applications, Lecture Notes in Electrical Engineering series, 2009

1.3.4 Revue Internationale (9)

RI09: « MemExplorer: From C code to Memory Allocation », Johann Laurent, André Rossi, and Marc Sevaux, Journal of Low Power Electronics, Vol 8, 2012

RI08 : «Spatial Switching data coding technique analysis and improvements for interconnect power consumption optimization», Antoine Courtay, Johann Laurent, and Olivier Sentieys, Journal of Low Power Electronics, Vol 6, 2010

RI07: «Energy and power consumption estimation for embedded applications and operating systems », Saadia Dhouib, Eric Senn, Jean-Philippe Diguet, Dominique Blouin, Johann Laurent, Journal of Low Power Electronics vol 5, 2009

RI06 : « High-Level Interconnect Delay and Power Estimation », Antoine Courtay, Olivier Sentieys, Johann Laurent, and Nathalie Julien, . Journal of Low Power Electronics, Vol4, 2008

RI05 : «Power Consumption and Perfor mance's Library on DSPs: Case Study MPEG2», Jalel Ktari, Mohamed Abid, Nathalie Julien, Johann Laurent, Journal of Computer Science, Vol 3, 2007

RI04 : «SoftExplorer: Estimating and Optimizing the Power and Energy Consumption of a C Program for DSP Applications», Johann Laurent, Eric Senn, Nathalie Julien, and Eric Martin, Eurasip Journal on Applied Signal Processing, Vol16, 2005

RI03 : « Power Consumption Modeling of the TI C6201 and Characterization of its Architectural Complexity », Johann Laurent, Nathalie Julien, Eric Senn, Eric Martin, IEEE Micro, Special Issue on Power- and Complexity-Aware Design, 2003

RI02 : « Functional Level Power Analysis of Complex Processor », Johann Laurent, Nathalie Julien, Eric Senn, and Eric Martin, Global DSP, issue 9, 2003

RI01 : « High Level Power Analysis for Embedded DSP Software », Johann Laurent, Nathalie Julien, and Eric Martin, IEEE TCCA NewsLetter, January 2001

1.3.5 Revue Nationale (1)

RN01: « Méthodes et outils d'estimation de la consommation de code embarqué sur processeur », Johann Laurent, Eric Senn, Nathalie Julien, Technique et Science Informatiques (TSI), Vol 26, 2007

1.3.6 Conférence Internationale (26)

CI26 : «Electronic Navigational Chart Generator for a marine mobile augmented reality system », Jean-Christophe Morgère, Jean-Philippe Diguet, Johann Laurent, MTS/IEEE OCEANS 2014

CI25 : « Mobile Augmented Reality System for marine navigation assistance », Jean-Christophe Morgère, Jean-Philippe Diguet, Johann Laurent, IEEE International Conference on Embedded and Ubiquitous Computing, 2014

CI24 : « Coupled open navigation and augmented reality systems for skippers », Ronan Douguet, Jean-Christophe Morgère, Jean-Philippe Diguet, Johann Laurent, International conference on innovation in high performance sailing yachts, 2013

CI23 : « TAG SHEPERD: a Low Cost and Non Intrusive Man Overboard Detection System », Johann Laurent, Nicolas Le Griguer, Jean-Philippe Diguet, International conference on innovation in high performance sailing yachts, 2013

CI22 : « Open Data Buoy to Analyze Weather and Sea Conditions for Sailing Regattas », Ronan Douguet, Jean-Philippe Diguet, Johann Laurent, Yann Riou, MTS/IEEE OCEANS, 2013

CI21 : « A New Real-Time Method for Sailboat Performance estimation based on Leeway Modeling », Ronan Douguet, Jean-Philippe Diguet, Johann Laurent, Yann Riou, The 21st Chesapeake Sailing Yacht Symposium, 2013

CI20 : «Modelling and estimating the energy consumption of embedded applications and operating systems », Saadia Dhouib, Eric Senn, Jean-Philippe Diguet, Johann Laurent, IEEE 12th International Symposium on Integrated Circuits, 2009

CI19 : «Interconnect Explorer: A High-level Power Estimation Tool for On-Chip Interconnects », Antoine Courtay, Johann Laurent, Olivier Sentieys, Nathalie Julien, IEEE/ACM Design Automation Conference, 2009

CI18 : « Model Driven High-level Power Estimation of Embedded Operating Systems Communication Services », Saadia Dhouib, Eric Senn, Jean-Philippe Diguet, Johann Laurent, Dominique Blouin, IEEE International Conferences on Embedded Software and Systems, 2009

CI17 : « A Convolutional Code for On-chip Interconnect Crosstalk Reduction », Antoine Courtay, Emmanuel Boutillon, Johann Laurent, IEEE International Symposium on Circuits and Systems, 2009

CI16 : « Multi-Level power consumption modelling in the AADL design ow for DSP, GPP, and FPGA », Eric Senn, Johann Laurent, Jean-Philippe Diguet, International Workshop on Model Based Architecting and Construction of Embedded Systems, 2008

CI15 : « Refining power consumption estimations in the component based AADL design flow », Forum on specification & Design Languages, 2008

CI14 : « Novel Cross-Transition Elimination Technique Improving Delay and Power Consumption for On-Chip Buses », Antoine Courtay, Johann Laurent, Olivier Sentieys, Nathalie Julien, IEEE International Workshop on Power and Timing Modeling, Optimization and Simulation, 2008

CI13 : « Energy models of real time operating systems on FPGA », Saadia Dhouib, Jean-Philippe Diguet, Eric Senn, Johann Laurent, Euromicro Workshop on Operating Systems Platforms for Embedded Real-Time Applications, 2008

CI12 : « New Directions in Inter- connect Performance Optimization », Antoine Courtay, Johann Laurent, Nathalie Julien, and Olivier Sentieys ,3rd International Conference on Design and Technology of Integrated Systems in Nanoscale, 2008

CI11 : « Functional Level Power Analysis: An Efficient Approach for Modeling the Power Consumption of Complex Processors », Johann Laurent, Eric Senn, Nathalie Julien, and Eric Martin, IEEE/ACM Design Automation and Test in Europe, 2004

CI10 : « Algorithmic level power and energy optimization for DSP applications: SoftExplorer », Eric Senn, Johann Laurent, Nathalie Julien, Eric Martin, IEEE International Symposium Integrated Virtual Circuits, 2004

CI09 : « Power Consumption Estimation of a C Algorithm : A New Perspective for Software Design », Johann Laurent, Nathalie Julien, Eric Senn, Eric Martin, IEEE LCR conference, 2002

CI08 : « Power Consumption Estimation of a C Program for Data-Intensive Applications », Johann Laurent, Nathalie Julien, Eric Senn, Eric Martin, IEEE International Workshop on Power and Timing Modeling, Optimization and Simulation, 2002

CI07 : « Algorithmic Power Consumption Estimation : an Efficient Guide for the Co-Design », Johann Laurent, Nathalie Julien, Eric Senn, Eric Martin, IEEE/ACM Design Automation Conference, 2002

CI06 : « Power Estimation of a C Algorithm on a VLIW Processor », Johann Laurent, Nathalie Julien, Eric Senn, Eric Martin, IEEE WCED, 2002

CI05 : « Power Estimation of a C Algorithm based on a Functional Analysis of a Digital Signal Processor », Johann Laurent, Nathalie Julien, Eric Senn, Eric Martin, IEEE ISHPC, 2002

CI04 : « High Level Energy Estimation for DSP Systems », Johann Laurent, Nathalie Julien, Eric Senn, Eric Martin, IEEE International Workshop on Power and Timing Modeling, Optimization and Simulation, 2001

CI03 : « High Level Power Estimation based on a Functional Analysis for Embedded DSP », Johann Laurent, Nathalie Julien, Eric Senn, Eric Martin, IEEE/ACM International Workshop on Logic and Synthesis, 2001

CI02 : « High Level Power Estimation for DSP », Johann Laurent, Nathalie Julien, Eric Senn, Eric Martin, IEEE/ACM SAME conference, 2000

CI01 : « High Level Power Analysis for Embedded DSP Software », Johann Laurent, Nathalie Julien, Eric Senn, Eric Martin, IEEE/ACM MEDEA conference 2000

1.3.7 Conférence nationale (8)

CN08: « Allocation de mémoire dynamique dans les systèmes embarqués Maria Soto, André Rossi, Marc Sevaux, Johann Laurent, Congrès de la société française de recherche opérationnelle et d'aide à la décision, 2012

CN07: « Impact du type d'architecture sur la consommation d'une application », Johann Laurent, Philippe Coussy, Journées Faible Tension Faible Consommation, 2007

CN06: «Modélisation et estimation de la consommation des interconnexions dans les SOC», Antoine Courtay, Johann Laurent, Olivier Sentieys, Nathalie Julien, Journées Faible Tension Faible Consommation, 2007

CN05 : «Modélisation et estimation de la consommation des interconnexions dans les SOC», Antoine Courtay, Johann Laurent, Olivier Sentieys, Nathalie Julien, Journées Faible Tension Faible Consommation, 2007

CN04 : «Etude de la consommation de plates- formes multiprocesseurs», Johann Laurent, Eric Senn, Nathalie Julien, Eric Martin, Journées Faible Tension Faible Consommation, 2005

CN03 : «Estimation de la consommation logicielle dans un système embarqué : Etude de cas», Jalel Ktari, Johann Laurent, Mohamed Abid, Nathalie Julien, Journées Faible Tension Faible Consommation, 2005

CN02 : «Estimation de la consommation d'un algorithme C intégrant l'architecture cible : une aide efficace pour la conception système», Johann Laurent, Nathalie Julien, Eric Senn, Eric Martin, Journées Francophone d'Adéquation Algorithme Architecture, 2002

CN01 : «Estimation de la Consommation d'un Algorithme C par Analyse Fonctionnelle», Johann Laurent, Nathalie Julien, Eric Senn, Eric Martin, Colloque GDR CAO, 2002

1.4 Activités d'enseignement

Depuis mon recrutement en tant que Maître de Conférences, ma pratique de l'enseignement a beaucoup évolué grâce à la pratique. En effet, je privilégie aujourd'hui beaucoup plus la pédagogie par l'exemple qu'il y a quelques années où je restais souvent trop sur les concepts généraux. Mes enseignements sont devenus de fait plus interactif avec les étudiants, j'essaie également de leur montrer le plus souvent possible comment sont utilisées les compétences que je leur enseigne dans le monde industriel. Enseignant dans une filière professionnelle, j'aborde également beaucoup de compétences sous forme de projets réalisés en binôme ou en équipe. Ceci permet en plus de mettre en pratique leurs compétences techniques, de développer des compétences plus transverses comme le travail en équipe, l'autonomie et la gestion du temps.

Réalisant beaucoup de projets et de travaux pratiques, ma charge d'enseignement annuelle se situe souvent autour de 300h équivalent TD. La section suivante décrits les matières sur lesquelles j'interviens, le public ainsi que les objectifs de l'enseignement.

1.4.1 Capteurs & Information

Public : Licence 3 Physique Sciences de l'Ingénieur parcours Electronique

Volume Horaire : 4h de cours, 4h de TD, 4h de TP

Ce cours a pour but de mettre à niveau les étudiants sur les principes d'échantillonnage et de conversion analogique/numérique et inversement. Nous abordons ici quelques techniques de conversion A/N et N/A, comment bien choisir le convertisseur en fonction des besoins et des contraintes, quel capteur pour quelle application. Lors de la séance de travaux pratiques, nous abordons la récupération d'une information d'un capteur de distance via la programmation d'un CAN (Convertisseur Analogique Numérique) sur un microcontrôleur. Les aspects choix de convertisseurs et de capteurs sont mis en pratique lors du projet de conception que je leur fais réaliser en fin de 1er semestre.

1.4.2 Projet Conception de plateforme

Public : Licence 3 Physique Sciences de l'Ingénieur parcours Electronique

Volume Horaire : 2h de cours, 2h de TD, 28h de TP

Ce projet a pour but de mettre en pratique le cours de capteurs & information mais également de leur faire travailler des compétences non techniques. Je change le sujet de ce projet tous les ans mais le concept sous-jacent reste identique à savoir la réalisation d'un système complet du capteur à l'actionneur. Pour cela je les fais partir d'un cahier des charges présentant les fonctions à réaliser avec comme contraintes l'utilisation d'une carte microcontrôleur, la réalisation d'un petit PCB et le respect d'un coût maximal pour le système. Ils doivent alors construire leur solution en expliquant tous leurs choix et en les justifiant. Durant ce projet j'intègre une partie interactive avec eux en les soumettant à des questions afin de vérifier qu'ils ont couverts tous les aspects de la conception. Ce premier projet se focalise que sur le choix des composants et la réalisation d'un PCB sur des outils de CAO. La partie programmation est abordée au second semestre dans le cadre d'un autre projet que j'encadre également.

1.4.3 Asservissement linéaire

Public : Licence 3 Physique Sciences de l'Ingénieur parcours Electronique

Volume Horaire : 8h de cours, 8h de TD, 8h de TP

Ce cours aborde les aspects stabilité et compensations des systèmes. Dans un premier temps, j'aborde ce qu'est une fonction de transfert d'un système comment nous pouvons la déterminer et plus particulièrement l'utilisation des méthodes par essais. J'introduis alors les diagrammes de Bode et de Nyquist dans le but de leur faire analyser le comportement d'un système. L'idée ici est toujours de mettre en regard les aspects théoriques avec leurs représentations pratiques (implication d'une grande marge de gain par exemple etc…). Dans un second temps, j'aborde avec eux la correction d'un système afin de garantir un fonctionnement particulier ; nous voyons ici la correction par avance et retard de phase et les correcteurs P, PI & PID. Les TP mettent en lumière les différents points du cours abordés ; j'utilise pour cela les outils Scilab et Matlab.

1.4.4 Architecture des microprocesseurs

Public : Licence 3 Physique Sciences de l'Ingénieur parcours Electronique

Volume Horaire : 10h de cours, 8h de TD, 8h de TP

Ce cours est la première partie d'un cours qui s'étend sur le Master 1 et qui a pour but de leur faire comprendre le fonctionnement interne d'une architecture logiciel. J'aborde avec eux la notion d'automate à états finis afin qu'ils puissent comprendre le fonctionnement de la partie contrôle d'une architecture. Je ne vois dans ce cours que les architectures scalaires et non pipelines afin qu'ils puissent appréhender les concepts de base dans un premier temps. J'aborde également les architectures mémoires et plus particulièrement le fonctionnement des mémoires caches. Je fais également un petit aparté sur la réalisation d'une puce i.e. les procédés de fabrications microélectroniques car généralement ils ne savent pas ou peu comment les puces sont fabriquées. En travaux pratiques, ils utilisent des simulateurs d'architectures scalaires leur permettant ainsi de bien comprendre le fonctionnement interne (de la lecture d'une instruction au rangement du résultat de calcul). L'utilisation de logiciels libres leur permet de pouvoir utiliser ces simulateurs sur leur machine en dehors des heures encadrées.

1.4.5 Projet microprocesseur

Public : Licence 3 Physique Sciences de l'Ingénieur parcours Electronique

Volume Horaire : 0h de cours, 0h de TD, 36h de TP

Ce projet fait suite au projet de conception de plateforme du premier semestre. Dans cette partie, les étudiants reprennent le PCB qu'ils ont développé au premier semestre et viennent l'interfacer avec une carte microcontrôleur. Je leur fais utiliser un microcontrôleur STM32 de ST Microelectronics basé sur un cœur ARM Cortex M4. Les objectifs de ce projet sont multiples mais ont pour point commun le développement d'un système réactif temps réel. Je les oblige à développer dans un premier temps leur architecture logiciel sur le « papier » afin qu'ils aient une bonne vision de l'articulation des fonctions à mettre en œuvre. Le système devant être réactif, les étudiants ont la possibilité de soit utilisé un OS temps réel (Free RTOS par exemple) soit de développer tout leur système sous interruption. Je leur conseille la seconde solution car elle leur permet d'appréhender la gestion des priorités lors d'interruptions multiples et de limiter la « surcharge mémoire » due à l'OS.

1.4.6 Architecture avancée

Public : Master 1 Génie Electrique & Informatique Industrielle

Volume Horaire : 10h de cours, 8h de TD, 16h de TP

Ce cours fait suite à celui de Licence 3 et aborde les aspects avancés des architectures processeurs. J'aborde, dans un premier temps, les aspects pipeline afin de montrer aux étudiants comment ce concept permet d'améliorer les performances d'une architecture sans rajouter de parallélisme au niveau des opérateurs de calcul. Je leur présente ensuite les aspects de prédiction de branchement afin de montrer comment ce procédé peut permettre de limiter les aléas dus aux branchements. Enfin, j'aborde avec eux le parallélisme de calcul en leur présentant les machines VLIW (parallélisme statique) et les machines Superscalaires (avec exécution spéculative du code). Pour chacun de ces concepts, je leur fournis des exemples d'architectures et ils font également l'objet d'exercice lors

d'un TD. Enfin, une partie du cours est mis en œuvre lors des 4 séances de TP durant lesquelles je leur fait écrire du code pour une machine parallèle. Lors du premier TP ils écrivent une application succincte en assembleur pour une machine MIPS et ensuite ils utilisent un simulateur pour observer le bon fonctionnement de leur code et étudier ses performances. Durant la seconde séance, je leur fait écrire une application de filtrage type filtre FIR en prenant en compte l'architecture cible afin d'optimiser l'écriture de leur code et donc ses performances. Enfin les deux dernières séances servent à la mise en œuvre complète d'un filtre FIR sur une carte de développement DSP (TMS320C6416). Ils conçoivent le filtre en utilisant l'outil FDAtool de Matlab, écrivent le code C permettant la récupération d'échantillons via le codec audio de la carte, la réalisation du filtrage proprement dit et enfin l'écriture de la sortie du filtre via le codec audio afin de pouvoir écouter l'impact de leur filtre sur les échantillons d'entrée.

1.4.7 Séminaires

Public : Master 2 Génie Electrique & Informatique Industrielle

Volume Horaire : 4h de cours, 0h de TD, 0h de TP

J'interviens sur 2 séances de séminaire, la première sur l'utilisation d'architectures de type GPU (Graphical Processing Unit) pour réaliser des applications autres que celles pour lesquelles ces processeurs ont été développés. Je leur présente dans un premier temps l'architecture « classique » d'un GPU puis comment nous pouvons les programmer pour réaliser des applications massivement parallèles. La seconde séance porte sur la conception faible consommation d'un système embarqué. J'aborde quelles sont les questions à se poser pour un tel développement, comment estimer les besoins énergétiques et quelles sont les solutions qui peuvent être mises en œuvre sur des exemples concrets.

1.4.8 Projet de fin d'étude

Public : Master 2 Génie Electrique & Informatique Industrielle

Volume Horaire : 4h de TD par projet

Dans le cadre de ce projet de fin d'étude, l'idée est de confronter l'étudiant à la réalisation d'un besoin client. En effet, je fournis à l'étudiant un cahier des charges léger en donnant seulement les fonctionnalités que je veux pour l'application (ex : commande vocale de drone quadricoptère). L'étudiant doit alors faire une étude de faisabilité et proposer sa ou ses solutions lors d'une réunion d'avancement. A partir de sa présentation je pose des questions pour voir s'il a appréhendé l'ensemble du problème et s'il maitrise sa solution technique. Généralement, plusieurs réunions sont nécessaires afin d'aboutir à une solution viable du problème ; ceci leur permet de « toucher du doigt » la difficulté à concevoir un système à partir d'un besoin utilisateur. Une fois la solution validée, l'étudiant s'attèle à sa réalisation pratique sur une période de 6 semaines temps plein. Durant cette période, nous faisons un point d'avancement toutes les semaines afin de les mettre en situation similaire à celle qu'ils pourraient avoir en entreprise et cela me permet aussi de vérifier l'avancement dudit projet.

En plus de ces différentes activités, j'encadre également des stages au niveau de la licence 3, du master 1 et du master 2 GEII.

Chapitre 2

Estimation de la consommation des architectures logiciel

Résumé :

Ce chapitre aborde la problématique de l'estimation d'énergie haut niveau des architectures logiciel embarquées. Ces travaux se sont faits dans le cadre du projet européen ITEA SPICES dont le but était de fournir une plateforme d'estimation d'un système embarqué critique. Nous avons donc proposé une méthodologie permettant la modélisation en consommation d'un système d'exploitation temps réel. Ce modèle peut être utilisé à plusieurs niveaux d'abstraction afin d'être intégré dans le flot de conception haut niveau OSATE basé sur le langage AADL (Architecture Analysis and Design Langage). Ce flot est utilisé par le partenaire principal du projet, à savoir, AIRBUS et permet de vérifier les spécifications fonctionnelles, temporelles et énergétique du système complet dès les premières phases de conception. Toute cette approche a été intégrée dans un outil CAT (Consumption Analysis Tool) qui, aujourd'hui, est fourni dans le flot d'OSATE.

2.1 Introduction

Les contraintes de consommation d'énergie et de puissance des systèmes électroniques ne cessent de croître. Afin de maitriser l'augmentation de leur consommation, il est nécessaire de modéliser et d'estimer la consommation d'énergie de chaque composant dès les premières phases de conception. Le projet SPICES (Support for Predictable Integration of mission Critical Embedded Systems) dans lequel s'inscrit ces travaux, propose de construire, à partir d'une description haut niveau, une implantation d'un système embarqué critique basée sur des composants prédictibles CCM (Corba Component Model). L'un des objectifs majeurs de SPICES est d'étendre le langage de description haut niveau AADL de manière à développer des méthodes et outils de CAO permettant de prédire les performances et la consommation d'énergie dès le stade de la spécification.

Un système embarqué est constitué de parties logiciels et matériels. La complexité croissante de ces systèmes entraine la nécessité de faire appel à un système d'exploitation afin de s'abstraire des couches basses de l'architecture ; l'idée est de facilité le développement d'applications complexes. Le système d'exploitation est un composant logiciel qui fournit des services aux tâches dont il permet l'exécution. Ces services peuvent être par exemple : la réalisation des communications entre tâches, l'ordonnancement de l'exécution de celles-ci, etc. Aussi le système d'exploitation est vu comme un gestionnaire des ressources du système. Il peut gérer la ressource processeur grâce aux algorithmes d'ordonnancement de tâches, la ressource mémoire grâce aux fonctions d'allocation et de libération de la mémoire, ainsi que toutes les autres ressources grâce à des gestionnaires de périphériques. Avec toutes ces fonctionnalités offertes par le système d'exploitation, ce dernier devrait avoir une contribution non négligeable à la consommation de puissance et d'énergie de l'architecture. Par conséquent, il est nécessaire de modéliser non seulement la consommation des mécanismes internes du système d'exploitation, mais aussi la consommation des composants matériels qu'il gère tels que le processeur, la mémoire et les périphériques d'entrée/sortie.

2.2 Analyse et modélisation de la consommation des systèmes d'exploitation (cas des systèmes embarqués)

Plusieurs travaux antérieurs ont tenté d'analyser, modéliser et même optimiser la consommation d'énergie des systèmes d'exploitation. Dick et al. [4] ont analysé l'énergie dissipée par un système

d'exploitation temps réel. Ils ont développé un outil de simulation et d'analyse d'énergie pour un système embarqué contenant un processeur SPARClite et le système d'exploitation μC/OS.

Acquaviva et al. [2] ont caractérisé la consommation d'énergie du système d'exploitation eCos. Ils ont mis l'accent sur la relation entre la consommation d'énergie d'eCos et la fréquence du processeur. Ils ont aussi analysé la consommation d'énergie due aux périphériques d'entrées/sorties.

Tan et al. [6] ont construit un modèle de consommation d'énergie d'un système d'exploitation temps réel. Ils ont testé leur approche sur deux systèmes d'exploitation à savoir μC/OS et Linux (arm linux v2.2.2).

Vahdat et al. [8] ont proposé de revoir plusieurs aspects du système d'exploitation pour minimiser la consommation due aux accès disque, à l'ordonnancement sur le processeur, aux allocations mémoire, et aux communications réseau. Ils ont tenu compte des différents modes (basse consommation) des composants matériels du système pour optimiser la consommation d'énergie.

Dans la suite, nous détaillons deux travaux : Tan et al [6] qui étaient les premiers à construire un modèle de consommation de l'OS, et Acquaviva et al [2] qui ont mis en évidence la consommation due aux périphériques d'entrée sortie.

La première approche que nous présentons a été développée à l'université de Princeton par Tan et al [6]. Les auteurs ont établi un macro modèle de la consommation d'un système d'exploitation temps réel. La méthodologie de caractérisation de l'énergie de l'OS est composée de quatre étapes (Figure 1). La première étape identifie les composants du système d'exploitation embarqué à caractériser. Après l'identification de ces composants, des programmes de test sont générés afin d'isoler les composants les uns des autres. Ensuite les programmes sont compilés et liés avec le système d'exploitation et exécutés dans un outil de simulation bas niveau [7].

L'énergie du système d'exploitation est divisée en 2 groupes : l'énergie explicite qui est liée aux primitives de l'OS (appels système) et l'énergie implicite qui n'est en rapport avec aucun appel système. L'énergie implicite résulte du fait que le système d'exploitation est en train de s'exécuter (énergie due aux interruptions du temporisateur, à l'ordonnancement, aux changements de contexte, aux traitements des signaux, etc). Les auteurs ont établi un macro modèle de consommation pour chacun de ces composants implicites ou explicites. Afin de mesurer les caractéristiques de l'énergie explicite, des appels successifs aux primitives du système d'exploitation ont été testés. Concernant l'énergie implicite, des programmes de test ont été élaborés afin de ne cibler que le composant voulu (ordonnancement, changement de contexte, etc.). Après avoir codé tous les programmes de test, vient l'étape de simulation. La construction du macro modèle se base sur une technique de régression. Les données d'énergie sont collectées en utilisant le simulateur. Etant donné cette collection de données, un modèle matriciel est formé. Deux systèmes d'exploitation embarqués ont été caractérisé : μC/OS et Linux OS. Les résultats des mesures ont montré un taux d'erreur variant entre 0.5% et 1.9% par rapport aux mesures obtenues par le simulateur. L'inconvénient de cette approche est qu'elle n'a pas traité la modélisation de la consommation des gestionnaires de périphériques et donc ignore la majeure partie de la consommation d'un système embarqué.

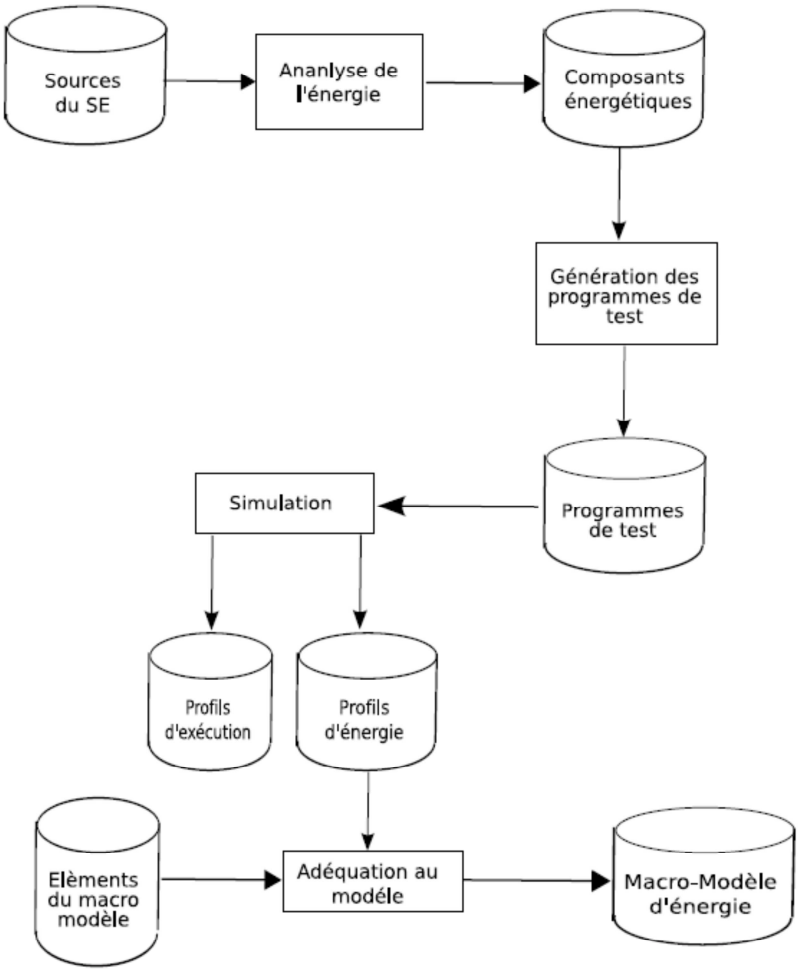


Figure 1: Méthodologie de caractérisation de l'énergie de l'OS

La seconde approche [2] que nous présentons caractérise la consommation d'énergie du système d'exploitation au niveau du noyau et des gestionnaires de périphériques. Le système d'exploitation étudié est eCos. En particulier, ils ont analysé la variation de la quantité de données d'entrées/sorties envoyée et la variation de la fréquence du changement de contexte.

L'approche proposée analyse les services du noyau du système d'exploitation et évalue la consommation d'énergie des gestionnaires de périphériques. Concernant le mécanisme de changement de contexte, les résultats ont montré que pour une fréquence de changement de contexte donnée, la consommation d'énergie décroit lorsque la fréquence d'horloge croit. Ceci s'explique par le fait que lorsque la vitesse du processeur augmente, le temps d'exécution diminue, par conséquent le nombre de changement de contexte décroit et le temps total dû aux appels système décroit. Etant donné que l'énergie est le produit de la puissance moyenne et du temps d'exécution total, l'énergie consommée décroit en fonction de la diminution du temps d'exécution. Concernant les autres appels système du noyau, les expériences ont montré que plus on augmente la vitesse d'horloge, plus la consommation d'énergie diminue. Ceci est dû à la diminution de la consommation des composants statiques. Par conséquent, pour une application qui manipule une

petite quantité de données et qui fait rarement appel aux périphériques, il est judicieux de travailler avec une fréquence d'horloge élevée.

Après avoir analysé la consommation des routines et mécanismes du système d'exploitation, les auteurs se sont intéressés à la consommation des gestionnaires de périphériques. Ils ont testé la consommation du gestionnaire de périphérique audio. Selon la quantité de données envoyées à travers le gestionnaire de périphérique, la consommation d'énergie varie.

Finalement les expériences menées par les auteurs montrent que la consommation d'énergie est essentiellement due aux gestionnaires de périphériques et à la gestion de la concurrence d'accès aux ressources d'entrées/sorties.

Dans le tableau suivant, nous établissons une étude comparative des différentes approches de modélisation de la consommation d'un système d'exploitation pour système embarqué.

Modèle	*Services Étudiés*						*Analyse de la consommation d'énergie*	
	Appels système	*Communications inter processus*	*Changements de contexte*	*Ordonnancement*	*Gestion Mémoire*	*Gestion des périphériques d'E/S*	*Méthode*	*SE / µP*
R.P. Dick et al [1]	caractérisés	caractérisés	caractérisés	caractérisé	non caractérisé	non caractérisé	Mesures directes	µC/OS/ Sparclite
T.K. Tan et al [2]	modélisés	modélisés	modélisés	modélisé	non modélisé	non modélisé	Simulation	µC/OS/ Sparclite Linux/arm
A. Acquaviva et al [3]	caractérisés	non caractérisés	caractérisés	non caractérisé	non caractérisé	caractérisé (audio driver)	Mesures directes	eCOS/ StrongARM
K. Baynes et al [4]	caractérisés	caractérisés	caractérisés	caractérisé	non caractérisé	non caractérisé	Simulation	µC/OS Echidna NOS /Motorola MCORE
A. Weissel [5]	modélisation par rapport aux événements processeur/mémoire					modélisé	Mesures directes	Linux/ Intel XScale

Tableau 1: Comparaison entre les modèles de consommation des systèmes d'exploitation

D'après le tableau 1, nous constatons que les services de gestion de la mémoire et des périphériques n'ont pas été modélisés par la plupart des travaux antérieurs. Sauf pour [2] où le gestionnaire de périphérique audio a été étudié. Des mesures sur la consommation d'un système d'exploitation temps réel ont été aussi effectuées par S. Rouxel dans le cadre du projet SPICES. Les mesures ont été faites sur la plateforme ProtoBoard FF152. Le système d'exploitation testé est MicroC/OS II tournant sur le processeur Microblaze. Les mesures ont montré que la consommation de puissance des services du système d'exploitation est inférieure à la consommation de puissance totale de l'application. La consommation de puissance ne varie pas énormément d'un service à l'autre. Ces résultats montrent aussi qu'une application ayant un grand taux de parallélisme et qui tourne sans OS, consomme plus de puissance qu'en s'exécutant avec un OS car l'utilisation de l'OS tend à lisser le parallélisme intrinsèque de l'application.

2.3 Méthodologie de construction des modèles de consommation

Dans cette partie, nous détaillons la méthode adoptée pour la construction du modèle de consommation d'énergie du système en tenant compte du système d'exploitation. L'analyse des services du système d'exploitation se base sur l'API standard Posix [5]. Le choix s'est porté sur POSIX pour assurer la portabilité de l'étude sur différents systèmes d'exploitation. En premier lieu, nous présentons la plateforme qui va servir à exécuter les testsbench, à prendre les mesures d'énergie et à valider l'approche. En second lieu, nous expliquons la construction des testbenchs qui ont servi à établir le modèle de consommation des composants du système embarqué.

2.3.1 Plateformes de test

La construction du modèle de consommation d'énergie d'un système embarqué contenant un RTOS se fait à partir de l'analyse des services du système d'exploitation et de la mesure de la consommation de ces services en variant les paramètres applicatifs, intergiciels (paramètres du système d'exploitation) et architecturaux.

Une première étude sera mise en œuvre sur une plateforme de test (Figure 2) constituée d'une carte de développement XUP VirtexII-pro, une barrette de mémoire de 256 Mo (pour la carte de développement), une compact flash d'au moins 512 Mo, un câble série (liaison PC vers carte de développement), un câble Ethernet.

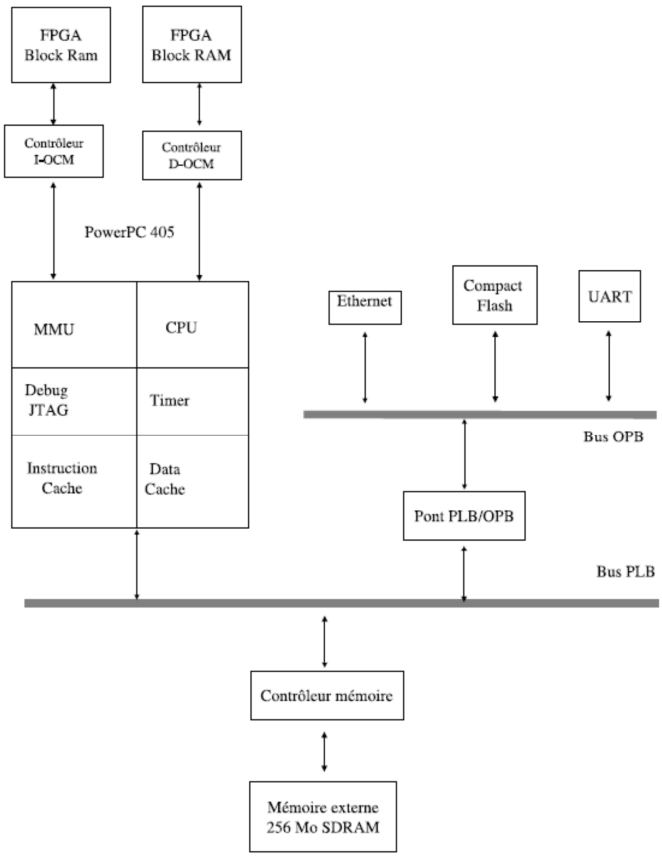


Figure 2: Système embarqué basé sur un processeur PPC-405

Dans le tableau 2 nous trouvons une étude comparative de différents systèmes d'exploitation temps réel. A l'issue de cette étude comparative et en tenant compte de notre plateforme matérielle, nous avons choisi d'utiliser (Montavista embedded Linux 3.1). Montavista tourne avec le noyau Linux 2.4.26, il est préemptif et intègre un ordonnanceur temps réel à priorités fixes. Cet ordonnanceur offre un nombre configurable de priorités temps réel et assure l'ordonnancement dans un système multiprocesseurs symétrique. Montavista a été choisi pour son aspect temps réel et parce qu'il supporte les périphériques spécifiques à Xilinx. (Xilinx SystemAce Controller, Ethernet MAC, UART Lite, GPIO ...).

RTOS	*Hard Real Time*	*LTT support*	*Posix compatibility*	*Sceduling Policy*	*Smal Memory Footprint (< 256KB)*	*Target*
RTlinux	Yes	Yes	Yes	prioritized FIFO scheduler	No (400KB-1MB)	X86, PowerPC, MIPS, Alpha
Montavista	Yes	Yes	Yes	Fixed priority RT scheduler	No (400KB-1MB)	PowerPC, ARM, MIPS; SuperH, Strong ARM, Xscale; xtensa
RTAI	Yes	Yes	Yes	preemptive priority based task scheduling	No (400KB-1MB)	X86, X86-64, PowerPC, ARM
µCLinux	No	No	Yes		No	microcontrollers without MMU (Motorola Dragon Ball, ETRAM, ARM7TDMI, intel i960, PRISMA, Microblaze,...)
eCOS	Yes (+ soft real time support)	No	Yes	-prioritized FIFO -Bitmap scheduling policy	Yes(10-100KB)	ARM, HitachiSH3, Intel x86, MIPS, PowerPC, SPARC, DSP

Tableau 2: Comparaison entre différents RTOS (Real Time Operating System)

2.3.2 Approche de modélisation de la consommation du système d'exploitation embarqué

Le but de l'approche est de construire un modèle de consommation d'énergie d'un système d'exploitation embarqué. Ce modèle de consommation sera utilisé lors des premières étapes de la conception afin d'optimiser la consommation d'énergie des composants du système embarqué gérés par le système d'exploitation. A partir de la description de l'architecture, du système d'exploitation et de l'application, on doit pouvoir estimer la consommation d'énergie du système (Figure 3).

La première étape de l'approche vise à établir une relation entre la manière dont le système d'exploitation va faire appel au processeur, mémoire, périphériques et la quantité d'énergie dissipée par ces composants. Les interactions entre l'application, le système d'exploitation et les composants matériels sont expliqués dans la Figure 4.

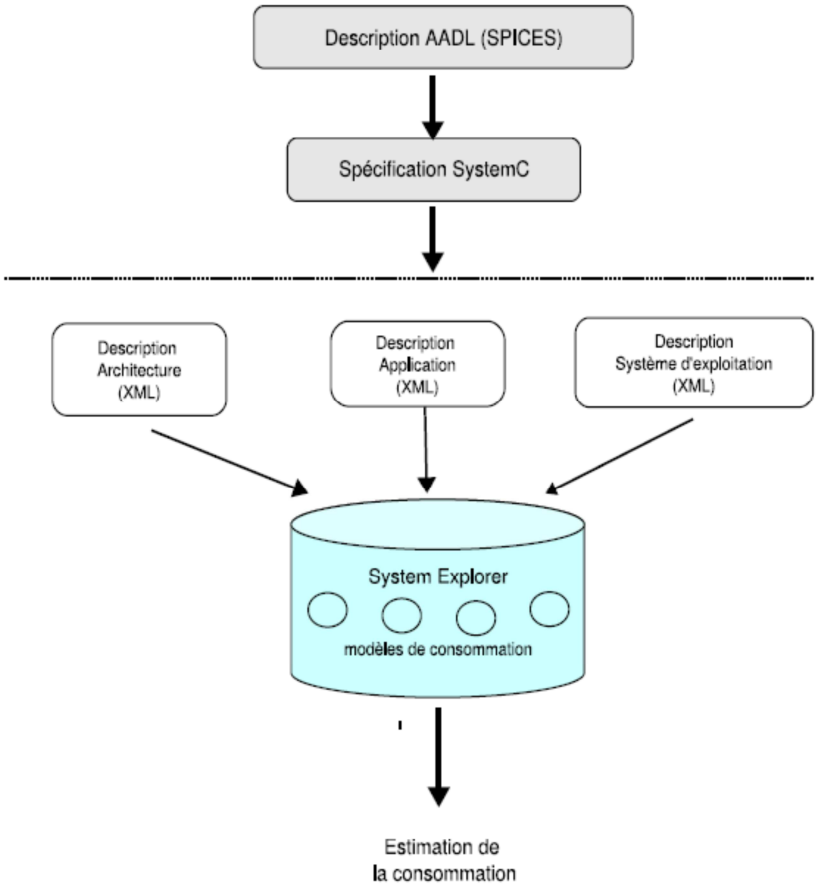


Figure 3: Estimation de la consommation d'un système embarqué au niveau système

2.3.2.1 Interactions application/système d'exploitation

Dans cette partie, nous détaillons les transitions d'états d'un processus. Ces transitions sont effectuées à la suite d'un évènement du système d'exploitation. Une fois qu'on a identifié toutes les transitions d'états d'un processus, on doit quantifier l'énergie dissipée à chaque transition.

Etats et transitions d'un processus : Nous nous plaçons dans le cas d'un système utilisant un mécanisme de swap pour gérer la mémoire. La figure 5 explique les états et transitions d'un processus. Le diagramme de transitions d'état permet de d´écrire l'ensemble des états possibles d'un processus. Il est clair que tout processus ne passera pas nécessairement par tous ces différents états. Les états d'un processus sont les suivants :

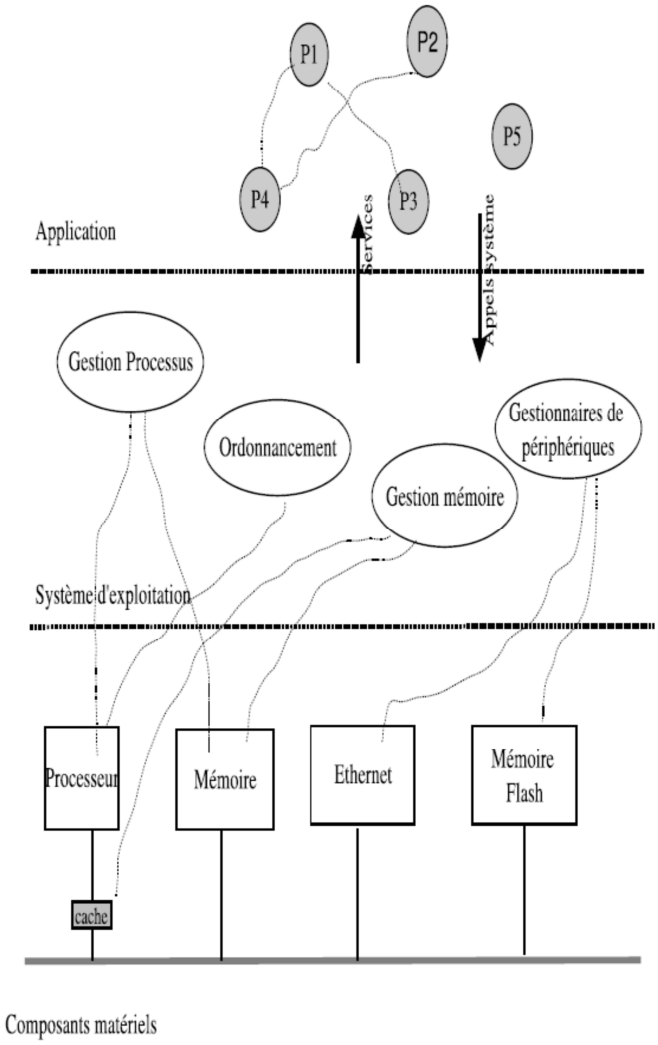


Figure 4: Interactions entre l'application, le système d'exploitation et les composants matériels

1. Le processus s'exécute en mode utilisateur.

2. Le processus s'exécute en mode noyau.

3. Le processus ne s'exécute pas mais est éligible (prêt à être exécuté).

4. Le processus est endormi en mémoire centrale.

5. Le processus est prêt mais le swappeur doit le transférer en mémoire centrale pour le rendre éligible.

6. Le processus est endormi en zone de swap (sur disque par exemple).

7. Le processus passe du mode noyau au mode utilisateur mais est préempté et a effectué un changement de contexte pour élire un autre processus.

8. Naissance d'un processus, ce processus n'est pas encore prêt et n'est pas endormi, c'est l'état initial de tous processus sauf le swappeur

9. Zombie : le processus vient de réaliser un exit, il apparait uniquement dans la table des processus où il est conservé le temps pour son processus père de récupérer le code de retour et d'autres informations de gestion (coût de l'exécution sous forme de temps, et d'utilisation des ressources).

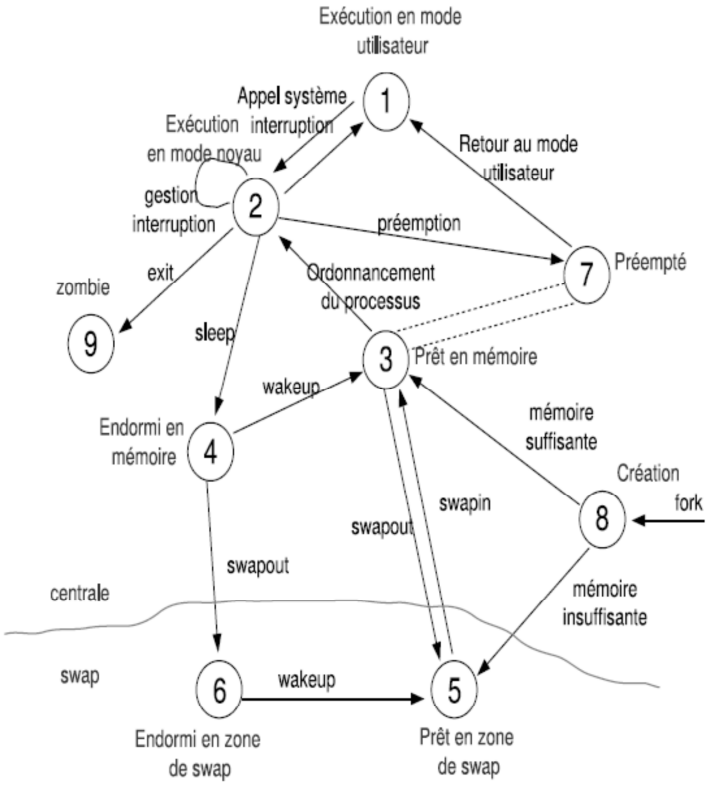


Figure 5: Diagramme d'états d'un processus

Le processus a un contrôle sur un nombre réduit de transitions : il peut faire un appel système, réaliser un exit, réaliser un sleep, les autres transitions lui sont dictées par le noyau selon des règles bien précises. Une de ces règles est par exemple qu'un processus en mode noyau ne peut être préempté. Certaines de ces règles sont définies par l'algorithme d'ordonnancement. Lors de l'exécution d'un programme, les transitions peuvent être tracées avec un outil appelé LTT (Linux Trace Toolkit) [10]. LTT permet l'enregistrement et l'analyse du comportement du système d'exploitation. Cet outil a un faible overhead et fournit une trace des évènements du noyau du système d'exploitation. Un autre aspect à caractériser est l'ensemble des communications inter processus. En effet, il y a plusieurs mécanismes de communication inter processus dans un système d'exploitation. Chaque mécanisme a ses spécificités et peut induire une consommation d'énergie différente par rapport à un autre mécanisme.

Communications inter processus : Il existe 3 types de mécanismes de communication inter-processus. Ces 3 IPC sont :

– Les queues de messages : Sortes de boites aux lettres où les processus peuvent déposer ou récupérer des messages

– La mémoire commune : Allocation d'une zone de mémoire partageable entre processus. Il n'y a pas de transfert d'informations. Le segment est directement adressable par le processus.

– Les sémaphores : Ensembles de compteurs mis à jour par des processus pour gérer les accès à des ressources communes. Les sémaphores ne sont pas à proprement parler des mécanismes de communication au sens large, mais des indicateurs comptabilisant les accès à des ressources partagées, et, en particulier, aux autres types d'IPC.

Au niveau applicatif, il faut donc modéliser la consommation d'énergie engendrée par chaque transition d'état d'un processus. Ceci implique la modélisation de la consommation de chaque transition en tenant compte du contexte du processus, c'est à dire, la taille des zones de données et de code du processus. Aussi, il faut modéliser la consommation de chaque mécanisme de communication inter processus.

2.3.2.2 Communications inter processus

Le système d'exploitation permet de :

- Ordonnancer l'exécution des processus sur le processeur
- Gérer les accès aux périphériques d'entrée sortie
- Assurer le transfert des données entre mémoire centrale et la mémoire secondaire
- Traiter les interruptions matérielles

Il faut modéliser la consommation d'énergie de la politique d'ordonnancement, des accès aux périphériques, de la politique de gestion de la mémoire et le traitement des interruptions matérielles.

Caractérisation des appels système : La consommation d'énergie des appels système peut être caractérisée par un appel répétitif à la fonction dans un programme. L'énergie consommée correspond à la transition entre 2 états d'un processus. Les appels système qui vont être caractérisés sont classés en cinq catégories : les appels pour la gestion de processus, les appels pour la communication inter processus, les appels pour les allocations mémoire, les appels pour les accès mémoire flash, et les appels pour les accès réseaux.

Caractérisation des services :

- **Changement de contexte** : Pour isoler l'énergie due aux changements de contexte, nous avons codé un programme contenant deux tâches qui se relaient sur le processeur. Pour avoir que le surcoût d'énergie dû aux changements de contexte, les données manipulées par les deux tâches sont de petite taille de façon qu'il n'y ait pas de défauts de cache.
- **Ordonnancement** : L'énergie dissipée due à l'ordonnancement est difficile à isoler, vu que l'ordonnancement est étroitement lié avec les changements de contexte et les interruptions. Par conséquent, pour avoir l´énergie due à l'ordonnancement, il faut déduire l'énergie due aux changements de contexte et aux interruptions du timer.
- **Gestion mémoire :** Pour mesurer la dissipation d'énergie due au swap, il faut créer un ensemble de processus de taille totale supérieure à la taille de la mémoire principale. Dans ce cas, le swap est nécessaire pour assurer la création de tous les processus.

2.4 Exemple de modèles de consommation : cas du tick timer

L'ensemble des modèles de consommation sont construits en utilisant la méthode FLPA (Functional Level Power Analysis) développée durant mes travaux de thèse.

L'interruption liée au timer est un élément important dans la gestion du système d'exploitation. Dans le cas du noyau 2.6 de Linux, le timer exécute périodiquement les tâches suivantes :

- Met à jour la disponibilité du système
- Met à jour la date du système
- Vérifie si le processus courant a consommé toute sa ressource temporelle et si oui provoque un ré ordonnancement
- Exécute tous les timers dynamiques qui ont expiré
- Met à jour l'utilisation des ressources et les statistiques d'utilisation du processeur

2.4.1 Méthode de modélisation utilisée

Afin de caractériser la consommation de puissance et d'énergie liée à l'interruption du timer système, nous avons exécuté des programmes avec et sans l'OS. Les programmes utilisant l'OS ont été exécutés avec des fréquences de tick timer différentes. Après avoir mesuré sur une carte de développement la consommation de l'exécution de chacun des programmes, nous calculons l'écart de consommation entre l'exécution avec et sans OS. Cette variation est ensuite mise en perspective avec la variation de la fréquence du tick timer afin de construire le modèle de consommation.

Sur un Linux 2.6, la fréquence du tick timer pour être programmée à 100, 250, 300 ou 1000Hz. La figure 6 présente l'évolution du courant consommé par l'exécution du processus idle sur la carte et ceci pour les 4 fréquences du timer. Nous pouvons voir que les pics de consommation correspondent à la fréquence du tick timer.

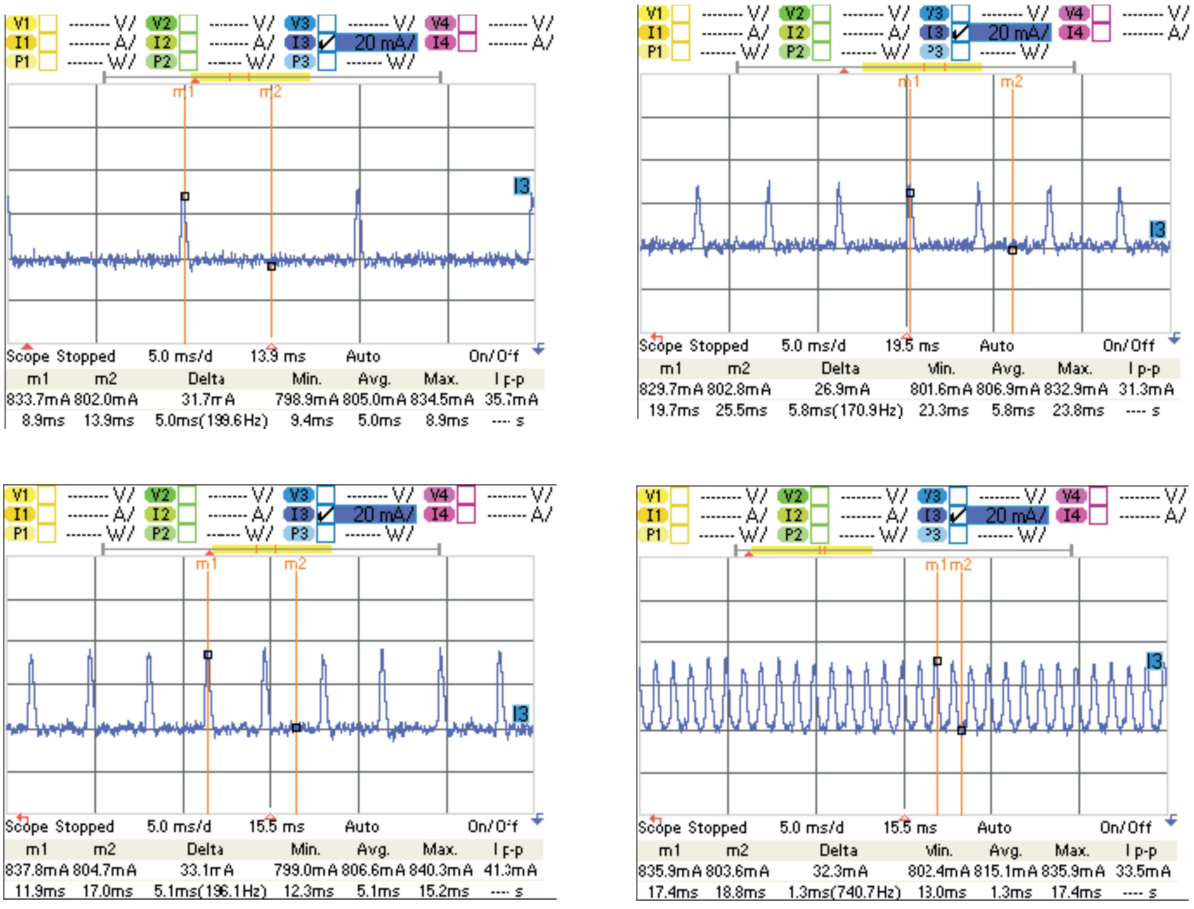


Figure 6: Evolution du courant pour l'exécution du processus idle et pour différentes fréquences du tick timer

Afin d'évaluer le surcoût en consommation engendré par l'interruption du timer, nous avons exécuté 2 programmes orientés flot de données avec et sans OS. Le premier programme réalise une DCT (Discrete Cosine Transform) sur un macro bloc de 8x8 pixels stocké en RAM. Le second réalise une quantification appliquée sur le résultat de la DCT. Chaque fonction est exécutée itérativement sur la même image afin de limiter les effets dus aux défauts de cache. Nous avons exécuté ces programmes sur différents processeurs et pour différentes fréquences de timer avec et sans OS.

La figure 7 présente les résultats de ces mesures et montre que le seul paramètre influant le surcoût du tick timer est sa fréquence.

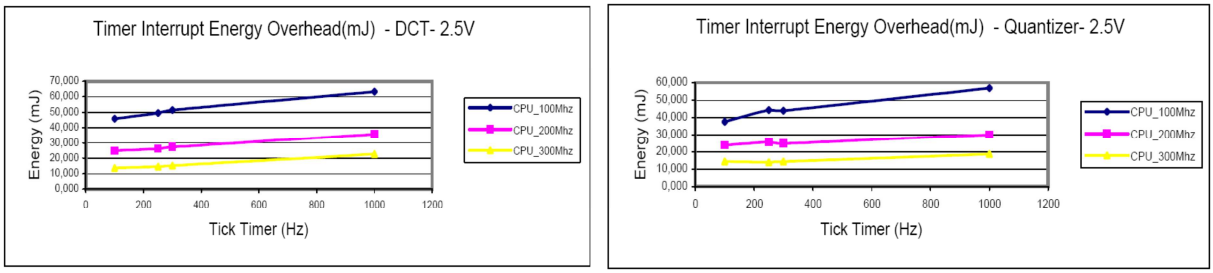


Figure 7: Surcoût d'énergie du timer en fonction de sa fréquence et du processeur

Le surcoût en consommation d'énergie est calculé comme suit :

$$\delta E_{timer\_interrupt} = E(T_{withOS}) - E(T_{withoutOS})$$

Avec T_{withOS} représente le temps d'exécution du programme avec un OS et $T_{withoutOS}$ le temps d'exécution du même programme sans l'OS.

Le surcout en consommation de puissance est calculé comme suit :

$$\delta(P_{timer}) = \frac{E(T_{withOS}) - E(T_{withoutOS})}{T(T_{withOS}) - T(T_{withoutOS})}$$

Où T est le temps d'exécution du programme mesuré lors des expériences. Nous avons constaté que les surcoûts en puissance et en énergie variaient de la même manière pour chacun des programmes et dépendaient de la fréquence du tick timer et du processeur. Le modèle de surcoût en puissance peut donc s'exprimer ainsi :

$$\begin{pmatrix} \delta P_1 \\ . \\ \delta P_2 \end{pmatrix} = \begin{pmatrix} \alpha_1 & \beta_1 \\ . & . \\ \alpha_n & \beta_n \end{pmatrix} \times \begin{pmatrix} X \\ 1 \end{pmatrix}$$

Où P_i correspond à la consommation de puissance due à la fréquence i du processeur, X est la fréquence du tick timer en Hz et α_i est un coefficient associé à la fréquence du tick timer.

Tous les autres modèles de consommation des services de l'OS sont construits en utilisant la même méthodologie. De plus amples détails sur cette méthodologie peuvent être trouvés dans l'article fourni en Annexe A.

2.5 Méthode d'estimation

Les modèles de consommation de puissance ainsi que la méthodologie d'estimation ont été intégrés dans un outil de CAD (Computer Aided Design) appelé CAT pour Consumption Analysis Tool. Cet outil combine les modèles de puissance avec un modèle d'architecture au niveau système afin de fournir une analyse de consommation à haut niveau. La base de l'outil CAT est un DSL (Design Specific Language) qui est défini afin de décrire des architectures de système dans une perspective d'analyse de la consommation de puissance. Ce DSL sert également à échanger de l'information entre notre outil et les autres outils d'analyse du flot OSATE.

La méthode d'estimation se passe en 3 étapes :

- Estimation de la consommation des tâches indépendamment les unes des autres en utilisant l'outil SofExplorer,
- Addition du surcoût de consommation lié aux services de l'OS à partir des modèles réalisés
- Addition du surcoût lié aux communications inter processus (IPC) à partir des modèles développés.

Plus de détails sur la méthode d'estimation sont fournis dans l'article en Annexe A.

Application de la méthode sur une application MJPEG

Nous allons appliquer la méthode d'estimation sur une application de codeur MJPEG (Motion JPEG). Cette application est exécutée sur une plateforme XUP V2 Pro de Xilinx qui comprend entre autre 256Mo de mémoire SDRAM, 512Mo de mémoire flash, une interface Ethernet ainsi que 2 cœurs de Power PC405 (hard cores). L'application est découpée en différents threads comme le montre la figure 8 ; chacun de ces threads se synchronise avec son ou ses successeurs via des signaux. Nous avons réalisé 3 scénarii différents : le premier n'utilise qu'un processus et plusieurs threads comme présenté ci-dessous, le deuxième utilise 2 processus qui communique à travers des mqueues enfin le dernier utilise 2 processus exécutés sur 2 cartes différentes et communiquant entre eux via un socket (Ethernet).

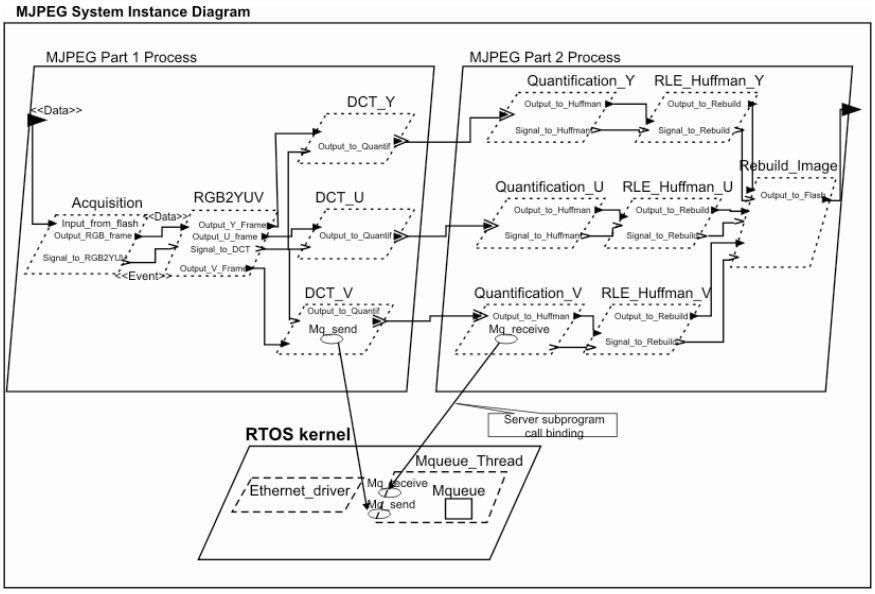


Figure 8: Découpage de l'application MJPEG en processus et threads

Nous ne présentons ici que les résultats pour le scénario 2. La spécification AADL nous permet de connaître le mapping de chaque tâche sur l'architecture. L'outil CAT commence par calculer la

consommation des tâches exécutées indépendamment en utilisant le modèle de puissance du PowerPC 405 développé durant mes travaux de thèse. Pour ce faire, l'outil CAT appelle les services de l'outil SoftExplorer (développé aussi durant ma thèse) qui analyse le code C associé à chacune des tâches. Les taux de défaut de cache sont extraits par profilage en utilisant l'outil Cachegrind fourni dans la suite d'outil Valgrind. Une fois cette étape réalisée, il faut rajouter le surcout lié aux services de l'OS tels que le tick timer et l'ordonnanceur. Enfin, nous n'avons plus qu'à additionner le surcoût lié aux communications inter processus à partir des modèles développés dans ses travaux.

L'estimation de la consommation d'énergie de l'application MJPEG pour le scénario 2 est d'environ 626mJ pour une énergie mesurée de 601mJ soit **4.16% d'erreur**.

2.6 Conclusion & réflexion

Ce chapitre a abordé la problématique de modélisation en consommation des services d'un système d'exploitation sur un système embarqué contraint. En couplant plusieurs approches, nous avons vu que nous pouvions obtenir des résultats d'estimation proche des valeurs mesurées sur des cartes de développement. Nous avons également montré qu'il était possible de réaliser ces estimations directement au niveau système en utilisant un langage de description de type AADL. Toutefois, ces travaux montrent aussi les limitations de ce type d'approche en effet, même si nous souhaitons réaliser les estimations à des niveaux toujours plus hauts, nous devons développer des modèles bas niveaux. La difficulté réside donc dans la possibilité ou non de déterminer à haut niveau les valeurs des paramètres des modèles comme par exemple, le taux de défaut de cache etc.

Une seconde limitation forte de ces approches est la conception même de ces modèles de consommation. En effet, pour réaliser des modèles avec une approche de type FLPA, mais ceci est aussi valable pour des approches de types ILPA, il faut une compréhension fine de l'architecture et être capable d'exciter les parties fonctionnelles de l'architecture indépendamment ou quasiment. Or aujourd'hui sur des architectures de type SoC ou MPSoC, les architectures internes deviennent extrêmement complexes du fait du parallélisme intrinsèque, des composants matériel utilisés (prédiction de branchement, cache multi niveaux, exécution spéculative du code etc.) ; il est alors très difficile voire impossible de maîtriser complètement le comportement de ce type de cibles.

Je pense qu'à l'avenir ce type de démarche de modélisation ne doit être utilisé que pour des cibles à complexité « réduite » et pour des plateformes qui seront utilisés sur une longue période dans le cadre de développement de produits. Ceci exclut bon nombre d'architectures actuelles et le marché des produits grand public dont l'obsolescence est un paramètre important voire rechercher. Pour ce type de cibles, je pense qu'il est plus judicieux de construire un pseudo modèle de consommation en utilisant des programmes de tests représentatifs des grandes classes d'applications. L'idée serait d'utiliser, par exemple, des applications de traitement d'image (H264…), de filtrage… de les exécuter sur la cible choisie et d'en mesurer la consommation d'énergie. A partir de ces mesures, réalisées sur un nombre restreint d'applications, nous pourrions alors construire une cartographie de la consommation de l'architecture cible. Lors de la phase d'estimation, l'idée serait alors de regarder à quelle classe d'application appartient celle que nous voulons estimer et de la placer dans notre cartographie de consommation afin d'obtenir une valeur d'estimation de sa consommation (cf. Figure 9). Cette estimation serait certes grossière mais dans bien des cas suffisante pour réaliser le dimensionnement du système à haut niveau.

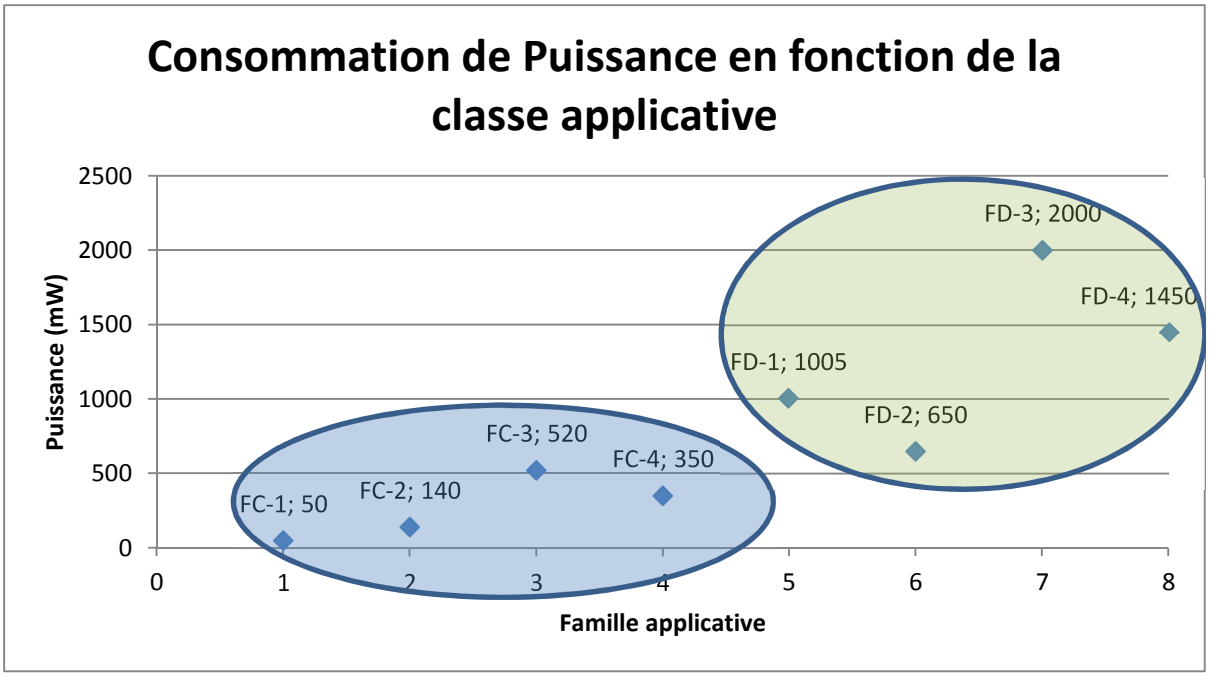


Figure 9: Consommation de puissance en fonction de la classe applicative

Références

[1] Modern operating systems. 2001.

[2] A. Acquaviva, L. Benini, and B. Ricc´o. Energy characterization of embedded real-time operating systems. In Proceedings of the Workshop on Compilers and Operating Systems for Low Power (COLP'01), sep 2001.

[3] S. Balakrishnan and J. Ramanan. Power-aware operating systems using ACPI, CS 736 project. Technical report, University of Wisconsin, Computer Science Department, 2001.

[4] R. P. Dick, G. Lakshminarayana, A. Raghunathan, and N. K. Jha. Power analysis of embedded operating systems. In Proceedings of the 37th Conference on Design Automation (DAC-00), pages 312–315, NY, jun 2000. ACM/IEEE.

[5] IEEE. IEEE Std 1003.1-2001 Standard for Information Technology — Portable Operating System Interface (POSIX) Rationale (Informative). IEEE, pub-IEEE-STD :adr, 2001. Revision of IEEE Std 1003.1-1996 and IEEE Std 1003.2-1992) Open Group Technical Standard Base Specifications, Issue 6.

[6] T. K. Tan, A. Raghunathan, and N. K. Jha. Embedded operating system energy analysis and macro-modeling. In Proceedings of the 2002 IEEE International Conference on Computer Design (ICCD'02), 2002.

[7] T. K. Tan, A. Raghunathan, and N. K. Jha. EMSIM : An energy simulation framework for an embedded operating system. In Proceedings of the IEEE International Symposium on Circuits and Systems ISCAS'02, 2002.

[8] A. Vahdat, A. Lebeck, and C. Ellis. Every joule is precious : A case for revisiting operating system design for energy efficiency. In Proceedings of the Ninth ACM SIGOPS European Workshop 2000, September 2000.

[9] A. Weissel. OS Services for Task Specific Power Management. PhD thesis, Erlangen University, 2006.

[10] K. YAGHMOUR AND M. R. DAGENAIS. MEASURING AND CHARACTERIZING SYSTEM BEHAVIOR USING KERNEL-LEVEL EVENT LOGGING. IN USENIX, EDITOR, 2000 USENIX ANNUAL TECHNICAL CONFERENCE : SAN DIEGO, CA, USA, JUNE 18–23, 2000, PAGES 13–26, PUB-USENIX :ADR, 2000. USENIX.

Chapitre 3

Estimation et optimisation de la consommation des interconnexions dans les SoC

Résumé :

Ce chapitre aborde la problématique de l'estimation et de l'optimisation de la consommation générée par le réseau d'interconnexion sur un système sur puce (System on Chip). En effet, le coût énergétique des interconnexions jusque là négligé lors de la conception d'un système devient, avec la complexité grandissante des puces et l'augmentation de la finesse de gravure, un des paramètres prépondérant du bilan énergétique. Les travaux effectués dans la thèse d'Antoine Courtay ont pour but d'améliorer la prise en compte de cette consommation. Pour se faire, il faut dans un premier temps modéliser la consommation liée à ces interconnexions en prenant en compte les aspects technologiques mais également les aspects liés à l'architecture du réseau comme par exemple l'apparition de diaphonie capacitive (appelée Crosstalk) lorsque les liens sont parallèles sur une longueur importante. Nous avons donc proposé un nouveau modèle de consommation en prenant en compte ces aspects. Dans un second temps, nous avons utilisé cette modélisation afin de proposer de nouvelles méthodes d'optimisation permettant un gain réel en consommation (prise en compte de la consommation du matériel rajouté). La méthode d'optimisation proposée a fait l'objet d'un brevet. Ces travaux ont également servi de point de départ d'une nouvelle thèse, que je co encadre avec André Rossi, sur l'optimisation de la consommation et de la latence pour les réseaux sur puce (Network on Chip).

3.1 : Introduction

Dans le développement des systèmes sur puce (System on Chip) actuel, les liens d'interconnexion qui servent à relier tous les composants de la puce entre eux posent différents problèmes. Le premier qui vient à l'esprit est la difficulté de router toutes ces interconnexions qui peuvent représenter plusieurs kilomètres si nous les mettons bout à bout. Ce problème est certes complexe mais n'est pas forcément le plus critique car il peut être résolu, ou en partie, par l'utilisation de technique de recherche opérationnelle. Le second est la consommation de puissance et d'énergie engendrée par l'activité sur ces liens de communication. En effet, chaque charge et décharge d'une ligne va engendrer des appels de courant et donc de la consommation. Lorsque le nombre de liens est faible cette consommation peut être négligée mais ce n'est plus le cas actuellement où le transport d'information devient le goulot d'étranglement de bon nombre d'application orientée flot de données. Enfin un dernier problème est l'apparition de délai sur les liens de communication dû au phénomène de diaphonie capacitive aussi appelé Crosstalk. Ce phénomène apparaît dès lors que des interconnexions vont être parallèles sur une longueur importante.

Les travaux de thèse réalisés par Antoine Courtay portaient donc sur 2 aspects : le premier consistait à proposer un modèle de consommation des interconnexions prenant en compte aussi bien les aspects technologiques que le phénomène de Crosstalk. Le second consistait à proposer de nouvelles méthodes d'optimisation de la consommation basées sur cette modélisation.

3.2 Modélisation physique du fil

Nos travaux de recherche sur la modélisation de la consommation des interconnexions se situent notamment au niveau physique afin d'obtenir des modèles de consommation et de délai très fiables. Les grandeurs physiques qui permettent de modéliser le comportement du fil sont au nombre de trois :

- R, la résistance du fil ;

- C, la capacité du fil;
- L, l'inductance du fil.

Ces grandeurs dépendent des caractéristiques du fil (sa composition métallique : cuivre, aluminium...) ainsi que de ses dimensions (hauteur, largeur, longueur). Nous allons nous intéresser à la consommation des interconnexions (les bus) qui véhiculent des informations entre différents blocs (par exemple entre un processeur et son cache) et non pas à la consommation des interconnexions telles que les lignes d'alimentation ou les arbres d'horloge.

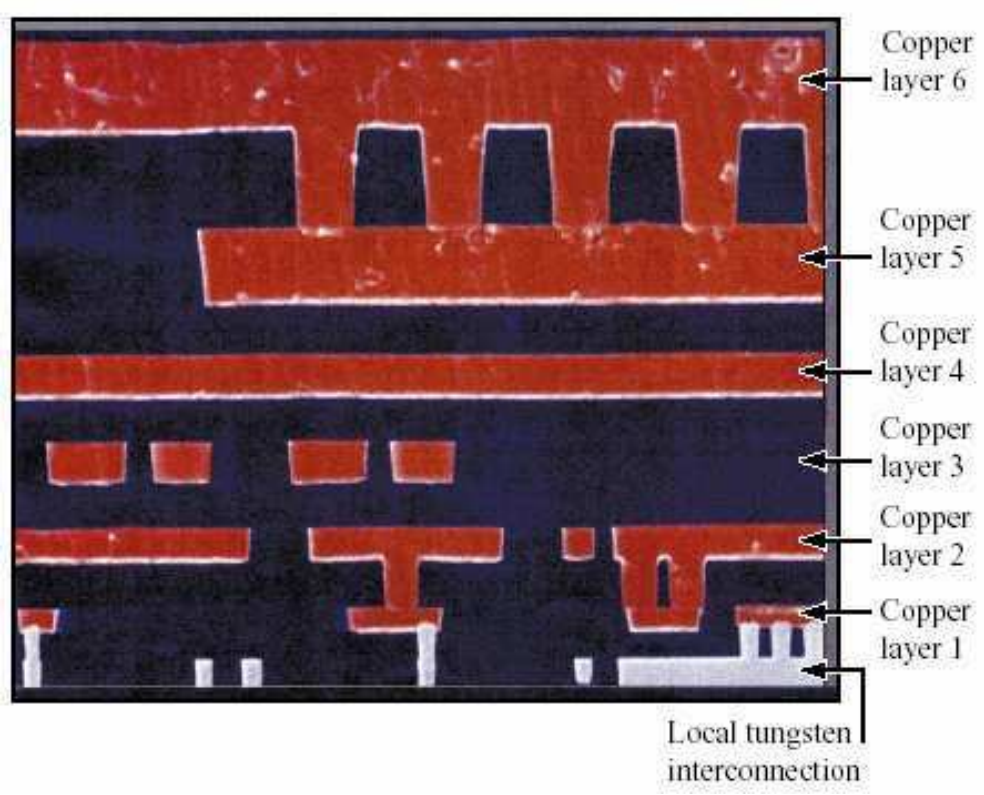


Figure 10: Réseau d'interconnexion vu en coupe

L'inductance (L), n'a d'importance que pour des technologies dites très submicroniques (≤ 45nm), pour des fils extrêmement longs où les variations de courant sont importantes et rapides et pour des fréquences de fonctionnement élevées; typiquement pour les lignes d'alimentation et la distribution de l'arbre d'horloge.

Pour des fils de taille raisonnable (jusqu'à quelques millimètres) propageant des signaux au sein d'un circuit, seules la capacité et la résistance sont significatives [1].Les grandeurs élémentaires permettant de caractériser le fil que l'on trouve dans les Design Kits des constructeurs sont au nombre de trois :

- $R_{\square} = \frac{\rho}{T}$, résistance par carré, exprimée en Ohm par carré [Ω/□] avec ρ la résistivité T du métal et T l'épaisseur du fil ;
- Csq, capacité élémentaire de la face inférieure du fil par rapport au substrat;
- Ce, capacité élémentaire des côtés du fil par rapport au substrat.

A l'aide de ces trois grandeurs, il est possible de calculer la résistance ainsi que la capacité du fil en fonction de ses dimensions (sa longueur L et sa largeur W exprimées en [m]). La figure 10 présente la

vue en coupe d'un réseau d'interconnexions en fonction des différents niveaux de métal ; il apparait alors évident que la capacité de la face inférieure du fil par rapport au substrat Csq ainsi que la capacité des côtés du fil par rapport au substrat Ce vont varier en fonction de la hauteur H de l'interconnexion par rapport au substrat et donc en fonction de la couche de métal utilisée. La résistance globale du fil est donnée par l'équation suivante :

$$R = R\square \frac{L}{W}$$

La capacité globale du fil est en fait la somme de deux capacités : la capacité globale de la partie inférieure du fil par rapport au substrat (parallel-plate capacitance) notée C_{pp} et la capacité globale des côtés du fil par rapport au substrat (fringing capacitance) notée C_f. Ces capacités sont représentées sur la figure 11. Les capacités C_{pp} et C_f sont données par les équations suivantes :

$C_{pp} = Csq.W.L$

$C_f = 2.Ce.L$

Le facteur 2 dans l'équation de Cf est destiné à inclure le fait que les deux côtés du fil contribuent à la capacité des bords. On obtient alors la capacité globale du fil par rapport au substrat qui vaut :

$Cs = L.[Csq.W + 2.Ce]$

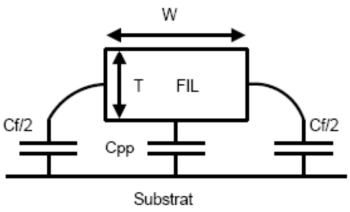


Figure 11: Détail des contributions des capacités d'un fil

Maintenant que l'on dispose des valeurs de ces grandeurs, il faut savoir quel modèle d'interconnexion va être utilisé ; c'est à dire qu'il faut savoir comment seront distribuées ces grandeurs afin de modéliser le mieux possible le comportement du fil. Les modèles d'inter-connexions peuvent être simples (modèle lumped) ou plus compliqués (modèles distribués) [2], ceci est expliqué plus en détail dans [1].

3.3 Modélisation physique du bus

Dans cette partie, les éléments qui entrent dans la composition d'un bus seront vus.

3.3.1 Modélisation physique des buffers

Les buffers (inverseurs) ont pour rôle de re générer le signal qui peut s'atténuer lors de la transmission sur la ligne. Les bus peuvent être bufférisés de différentes manières. Pour des lignes courtes où le signal se propage rapidement, on utilise en général deux buffers; un au départ de chaque ligne du bus (driver buffer) et un à l'arrivée (receiver buffer). Pour des lignes longues où le

signal s'atténue et où le temps de propagation devient critique, il est nécessaire de re générer le signal afin d'accélérer sa propagation en utilisant des méthodes d'insertion de buffer qui seront expliquées par la suite. Plus de détails sur la modélisation des buffers sont donnés dans [3].

3.3.2 Regroupement des lignes : le crosstalk

Généralement, les lignes d'interconnexions sont regroupées afin d'obtenir un bus de données permettant la transmission d'informations entre blocs. Le fait de regrouper les lignes d'interconnexions en un bus va faire apparaitre un nouveau phénomène de diaphonie capacitive que l'on nomme le crosstalk. La capacité de crosstalk entre deux fils adjacents dépend quant à elle de la surface en regard entre ces deux fils, et donc de l'épaisseur du fil, de sa longueur ainsi que de l'espacement entre ceux-ci :

$$C_c = \varepsilon_{ox}\frac{TL}{S}$$

- ε_{ox} représente la permitivité du diélectrique (oxyde) ;
- T représente l'épaisseur du fil (Thickness) ;
- L représente la longueur du fil (Length) ;
- S représente l'espacement entre deux fils (Spacing).

Lors de la transition des fils adjacents, il y a génération d'un bruit parasite sur le fil victime (i.e. : le fil central par rapport à ses voisins de gauche et de droite) dû au couplage entre les fils. Le bruit dû au crosstalk est relativement localisé. On modélise en général un système soumis au crosstalk en négligeant les ordres supérieurs au premier : ainsi on ne considère que trois fils comme le montre la figure 16. Le couplage capacitif entre les fils peut également se répartir sur les nœuds du modèle distribué Π3 qui a été présenté précédemment ; le modèle complet du bus qui est alors obtenu est représenté sur la figure 12. La partie suivante présentera les problèmes liés à l'évolution des paramètres résistifs et capacitifs et notamment les problèmes liés au crosstalk.

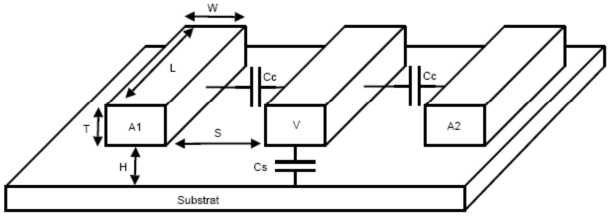


Figure 122: Un fil victime V soumis au couplage capacitif de ses deux agresseurs A1 et A2

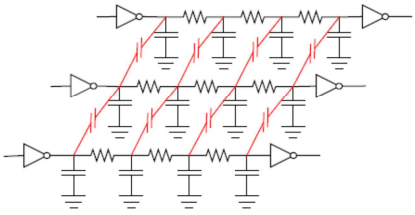


Figure 133: Modèle complet Π3 pour 3 fils avec couplage crosstalk

3.4 Evolution des paramètres technologiques

Cette partie va présenter comment se traduit l'évolution des paramètres technologiques (notamment les résistances et les capacités) et quelles en sont les conséquences sur les performances des interconnexions en termes de délai et de consommation (voir tableau 3).

Année	2001	2003	2005	2007	2009	2011	2013	2015
Technologie (nm)	130	100	80	65	50	40	32	25
Densité (Mtransitor/cm^2)	39	61	97	154	245	389	617	980
Frequence (MHz)	1684	2976	5204	9285	12369	17658	22980	33403
Tension d'alimentation (V)	1.2-1.1	1.2-1.0	1.1-0.9	1.1-0.8	1.0-0.8	1.0-0.7	0.9-0.6	0.9-0.6
Couches de métal	10	13	15	15	16	16	17	17
[1]Délai (ps)	x	191	307	486	783	1224	1572	5951
Puissance (W)	130	149	167	189	210	225	251	270
[2]Longueur routage (mm)	x	579	1019	1439	2000	2500	3125	4000
Constante diélectrique	3.0-3.6	3.3-3.6	3.1-3.4	2.7-3.0	2.5-2.8	2.5-2.8	2.1-2.4	1.9-2.2

Tableau 3: Evolution des paramètres technologiques tirés de [4], [5], [6]

Les prévisions de l'ITRS montrent une diminution de la finesse de gravure ainsi qu'une augmentation du nombre de transistors par puce. Ces augmentations corrélées avec une augmentation de la fréquence de fonctionnement des circuits ne peuvent qu'entraîner une forte augmentation de la puissance consommée.

Afin d'essayer de minimiser le phénomène de l'augmentation de consommation, on remarquera qu'un effort est porté sur la diminution de la tension d'alimentation. Avec la diminution de la technologie, les procédés de fabrication deviennent de plus en plus complexes et il devient de moins en moins facile de maîtriser les critères de délai et de consommation.

L'évolution des dimensions des transistors et des fils se traduit par une évolution du comportement du circuit, tout particulièrement au niveau temporel. Ainsi le délai d'un fil devient supérieur à celui d'une porte [7]. De plus, l'augmentation de la densité du réseau d'interconnexion ainsi que l'augmentation du nombre de couches de métal ne plaide pas en faveur de la réduction de la consommation et du délai. Dans les technologies plus anciennes (où la largeur de grille n'était pas inférieure à 250 nm), les interconnexions étaient réalisées en aluminium et séparées les unes des autres par un isolant (SiO2) dont la constante diélectrique avoisinait 4.

Avec l'évolution des procédés de fabrication, l'aluminium s'est vu remplacé par le cuivre dont la conductivité est bien meilleure. Grâce à cette évolution, le délai de propagation sur une interconnexion a fortement diminué. En parallèle à ce remplacement de l'aluminium par du cuivre, des isolants à faible permittivité (LowK) ont fait leur apparition pour se substituer au SiO2. Ces évolutions ont permis d'augmenter de façon significative les performances temporelles des interconnexions. L'atout principal de l'utilisation d'isolant à faible permittivité est de permettre de réduire les phénomènes de diaphonie capacitive entre les fils (crosstalk) ainsi qu'entre les fils et le substrat. Les deux sous parties suivantes présenteront comment se traduit l'évolution des paramètres résistifs et capacitifs sur le délai et la consommation des interconnexions.

3.4.1 Evolution des paramètres résistifs

Comme le montre la figure 14, le délai sur un fil de longueur constante entre différentes technologies ne cesse d'augmenter ; ce phénomène est dû entre autres à l'évolution de la résistance de ces fils. En effet entre deux technologies, les dimensions des fils diminuent et la résistance qui est inversement proportionnelle à la section du fil se voit augmentée.

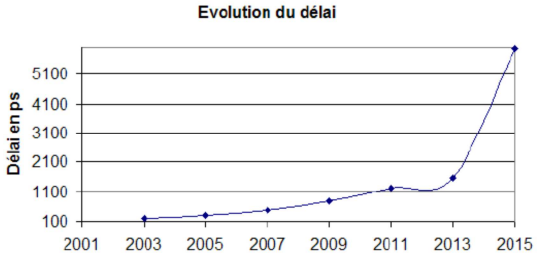


Figure 144: Evolution du délai (Métal1, longueur 1mm

3.4.2 Evolution des paramètres capacitifs

Les capacités parasites ne cessent d'augmenter ; en effet on peut noter que les dimensions physiques des fils ont changé ; les fils deviennent plus hauts que larges tels que le montre le tableau 4 ce qui favorise l'augmentation de la capacité de crosstalk entre les fils. Afin de compenser cette augmentation, des diélectriques à faible permittivité sont utilisés dans le but de réduire la capacité de couplage. La suite de cette partie expliquera plus précisément les problèmes liés au couplage capacitif.

Technologie (nm)	Largeur (μm)	Epaisseur (μm)	Rapport d'aspect (L/E)
180	0.8	1.25	1.56
130	0.6	1.2	2
90	0.5	1.2	2.4
65	0.45	1.2	2.66

Tableau 4: Evolution du rapport d'aspect sur les couches de métal globales (source [6])

Le crosstalk : source d'erreur

Comme il a été vu dans la section sur la modélisation physique du bus, il existe un couplage capacitif entre les fils. Lorsqu'une ligne commute, la ligne voisine est perturbée ; on définit alors la ligne perturbée comme la victime et la ligne perturbatrice comme l'agresseur. Il existe deux catégories de couplage :

- Le couplage positif, lorsque l'amplitude du bruit dépasse d'une valeur positive la tension sur le fil victime ;
- Le couplage négatif, lorsque l'amplitude du bruit dépasse d'une valeur négative la tension sur le fil victime.

Sur la figure 15, les différents types de couplage positifs et négatifs sont représentés :

- Sur la figure 15 a), l'agresseur effectue une transition montante alors que la victime reste à GND. Le cas sans couplage représenté en pointillés montre que la victime n'est pas perturbée. Avec couplage on observe un bruit positif au-dessus de GND ;
- Sur la figure 15 b), l'agresseur effectue une transition montante alors que la victime reste à Vdd. Avec couplage on observe un bruit positif au-dessus de Vdd ;
- Sur la figure 15 c), l'agresseur effectue une transition descendante alors que la victime reste à GND. Avec couplage on observe un bruit négatif en dessous de GND ;
- Sur la figure 15 d), l'agresseur effectue une transition descendante alors que la victime reste à Vdd. Avec couplage on observe un bruit négatif en dessous de Vdd.

Ce sont les cas a) et d) qui sont les plus ennuyeux ; en effet le pic de tension au-dessus de GND (pour le cas a)) et en dessous de Vdd (pour le cas d)) peut être source d'erreur dans le cas où ce pic parvient jusqu'à la tension de seuil du buffer en sortie du bus. Le bruit généré sur la ligne victime n'est pas le seul phénomène introduit par le crosstalk ; nous venons de voir le cas ici où le fil victime reste à un niveau logique stable pendant que son ou ses agresseurs commutent ; il va maintenant être expliqué ce qu'il se passe lorsque les fils agresseurs et le fil victime effectuent des transitions simultanées.

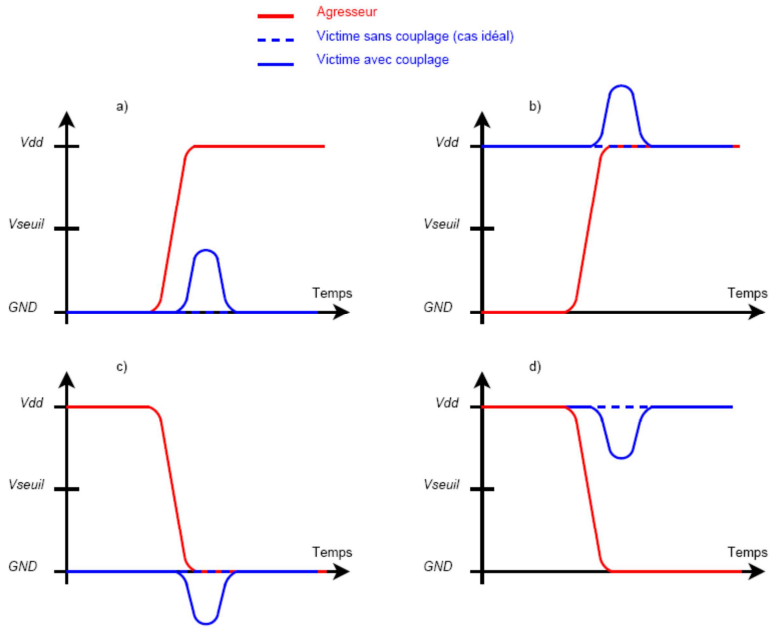


Figure 15: Bruits sur le fil victime (sans transition) en fonction du type de couplage

Le crosstalk : source de retard

Il a été expliqué dans la partie précédente qu'une transition sur un fil affecte son voisin en créant un pic de tension sur celui-ci. Lors de transitions simultanées sur le fil victime et agresseur, un pic de tension sera également généré ; ce pic peut selon la configuration des transitions légèrement accélérer ou considérablement ralentir la propagation de la donnée sur le fil victime (figure 16). On distingue également ici quatre cas de couplage :

- Sur la figure 16 a), l'agresseur et la victime effectuent des transitions montantes. Le cas sans couplage représenté en pointillés montre que les transitions se font identiquement. Avec couplage, la transition sur le fil agresseur va générer sur le fil victime un pic de tension dans le sens de la transition ; les transitions sur le fil victime et agresseur vont en quelque sorte s'accélérer mutuellement ;
- Sur la figure 16 b), l'agresseur effectue une transition montante alors que la victime effectue une transition descendante. Avec couplage, la transition sur le fil agresseur va générer sur le fil victime un pic de tension dans le sens inverse de la transition du fil ; la transition sur le fil victime va donc être ralentie ;
- Sur la figure 16 c), l'agresseur et la victime effectuent des transitions descendantes. Avec couplage, la transition sur le fil agresseur va générer sur le fil victime un pic de tension dans

le sens de la transition ; les transitions sur le fil victime et agresseur vont en quelque sorte s'accélérer mutuellement ;

- Sur la figure 16 d), l'agresseur effectue une transition descendante alors que la victime effectue une transition montante. Avec couplage, la transition sur le fil agresseur va générer sur le fil victime un pic de tension dans le sens inverse de la transition du fil ; la transition sur le fil victime va donc être ralentie.

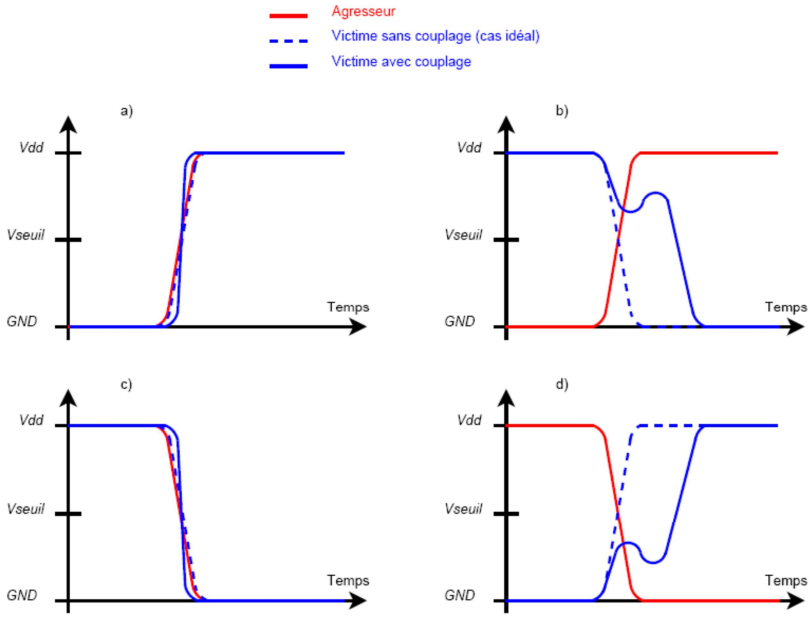


Figure 16: Bruits sur le fil victime (avec transition) en fonction du type de couplage

Lors des transitions, le fil victime va être parasité par les agresseurs ; ceux-ci peuvent être au nombre de un ou deux. En effet pour un fil situé au milieu du bus, ses voisins sont au nombre de deux, en revanche un fil situé aux extrémités du bus n'a qu'un voisin. Il est donc possible ici d'effectuer un classement des types de transition selon la rapidité de propagation des transitions. Ici ↑ représente une transition montante, ↓ représente une transition descendante et − signifie qu'il n'y a pas de transition sur le fil.

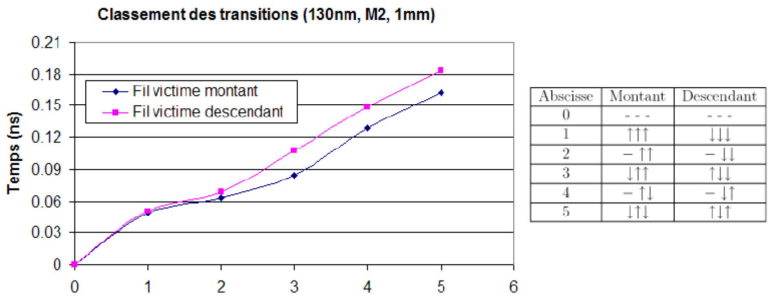
Prenons le cas de trois fils qui commutent dans la même direction (↑↑↑ ou ↓↓↓), la transition sur le fil victime (ie : le fil central) va se voir accélérée par les perturbations de ses deux agresseurs. Maintenant, si un de ses agresseurs ne transite pas (− ↑↑ ou − ↓↓), la transition sur le fil victime va se voir accélérée par la perturbation d'un seul de ses agresseurs et donc la transition sera moins rapide que dans le cas précédent.

Prenons maintenant le cas de trois fils où le fil victime et un de ses agresseurs commutent dans la même direction tandis que son autre agresseur commute dans le sens inverse (↑↑↓ ou ↓↓↑). Un de ses agresseurs va accélérer la transition et l'autre la ralentir. Maintenant si l'agresseur qui commute dans le même sens que la victime ne transite pas (− ↑↓ ou − ↓↑), la transition sur le fil victime va se voir ralentir par la perturbation d'un seul de ses agresseurs et donc la transition sera moins rapide que dans le cas précédent.

Poursuivons ce raisonnement en prenant le cas où les agresseurs commutent dans le sens inverse de la victime ↓↑↓ ou ↑↓↑), ici la victime va se voir perturbée par ses deux agresseurs et la transition

sera encore plus lente que dans le cas précédent ; ceci représente le pire cas de temps de propagation sur le bus.

La figure 17 illustre l'augmentation du temps de propagation sur le fil victime en fonction des transitions sur ses agresseurs sur une couche de métal réservée au bus pour une longueur typique de 1mm dans une technologie 130nm. (Les résultats expérimentaux sont les mêmes pour les autres technologies qui ont été expérimentées : 90nm et 65nm). On peut de cette manière effectuer un classement des transitions selon le temps de propagation sur le fil victime tel que le montre le tableau 6 où g représente le facteur de délai et r le ratio de la capacité de crosstalk par rapport à la capacité du fil par rapport au substrat. Outre les problèmes de bruit et de temps de propagation, le phénomène de crosstalk est également à l'origine de l'augmentation de la consommation, ce qui va être expliqué dans la partie suivante.



Abscisse	Montant	Descendant
0	- - -	- - -
1	↑↑↑	↓↓↓
2	− ↑↑	− ↓↓
3	↓↑↑	↑↓↓
4	− ↑↓	− ↓↑
5	↓↑↓	↑↓↑

Figure 17: Variation du temps de propagation sur le fil victime en fonction du type de transition

Le crosstalk : source de consommation

La consommation statique sur les bus peut être négligée puisque sa contribution dans la part de la consommation totale est très faible ; en effet les données qui circulent sur le bus changent souvent d'état ; C'est ce nombre important de transitions (donc une activité importante) qui va faire que la consommation sur les bus est exclusivement de la consommation dynamique. De plus, le nombre de transistors présents sur les bus est faible comparé à un bloc de calcul qui effectue des opérations plus complexes ; on aura donc peu de transistors qui vont fuir. Le calcul de la consommation dynamique se représente de la manière suivante :

$$P_{dynamique} = \sum_{i \in Nbit} \alpha_i . C_{L_i} . V_{dd} . V_{swing} . F$$

- N_{bit} représente le nombre de bits du bus considéré ;
- α_i représente l'activité du fil i ;
- C_{Li} représente la capacité du fil i (capacité qui va être détaillée ci-dessous) ;
- V_{dd} représente la tension d'alimentation ;
- V_{swing} représente la tension d'excursion ;
- F représente la fréquence des transitions.

Dans l'équation de la consommation dynamique, le paramètre CL se décompose en la somme de quatre capacités :

- C_{OUT} qui représente la capacité de sortie de l'inverseur à l'entrée du fil ;

- C_s qui représente la capacité du fil par rapport au substrat ;
- C_c qui représente la capacité de crosstalk ;
- C_{IN} qui représente la capacité d'entrée de l'inverseur à la fin du fil.

C_L	Types de transition				g
C_s	$(\uparrow,\uparrow,\uparrow)$	$(\downarrow,\downarrow,\downarrow)$			1
$C_s + C_c$	$(-,\uparrow,\uparrow)$	$(-,\downarrow,\downarrow)$	$(\uparrow,\uparrow,-)$	$(\downarrow,\downarrow,-)$	1+r
$C_s + 2.C_c$	$(-,\uparrow,-)$ $(\uparrow,\uparrow,\downarrow)$	$(-,\downarrow,-)$ $(\uparrow,\downarrow,\downarrow)$	$(\downarrow,\uparrow,\uparrow)$	$(\downarrow,\downarrow,\uparrow)$	1+2.r
$C_s + 3.C_c$	$(-,\uparrow,\downarrow)$	$(-,\downarrow,\uparrow)$	$(\uparrow,\downarrow,-)$	$(\downarrow,\uparrow,-)$	1+3.r
$C_s + 4.C_c$	$(\uparrow,\downarrow,\uparrow)$	$(\downarrow,\uparrow,\downarrow)$			1+4.r

Tableau 3: Capacité parasite (C_L) et facteur de délai (g) du fil victime en fonction du type de transition

On remarque donc que plus la capacité de crosstalk C_c est importante, plus la consommation va augmenter. Nous venons de voir que l'évolution des paramètres résistifs et capacitifs introduit une contrainte forte sur le temps de propagation des données au sein des interconnexions. C'est pourquoi lorsque ce temps devient trop critique, notamment pour des lignes d'interconnexions longues, les concepteurs de circuit ont recours à des méthodes d'insertion de buffers afin d'accélérer la propagation des données.

3.4.3 Evolution du délai : insertion de répéteurs

Cette partie va expliquer comment exprimer le temps de propagation au sein des interconnexions, ensuite les méthodes d'insertion de répéteurs seront présentées.

Expression du temps de propagation

Il existe deux méthodes de mesure du temps de propagation sur les interconnexions tel que le montre la figure 18.

- La première consiste à calculer la différence de temps entre le passage à 50% de la transition par rapport à sa valeur finale de la porte entrée et le passage à 50% de la transition par rapport à sa valeur finale de la porte en sortie ;
- La seconde consiste à regarder la différence entre le temps de passage de 10% à 90% par rapport à sa valeur finale de la transition de la porte en sortie (ou de 90% à 10% dans le cas d'une transition opposée).

Comme c'est très souvent la première définition du temps de propagation qui est retenue, c'est cette définition qui va être utilisée dans les méthodes d'insertion de buffers qui seront présentées par la suite. On trouve dans la littérature beaucoup de techniques ayant pour objectif de modéliser le temps de propagation d'une porte CMOS pilotant une charge capacitive.

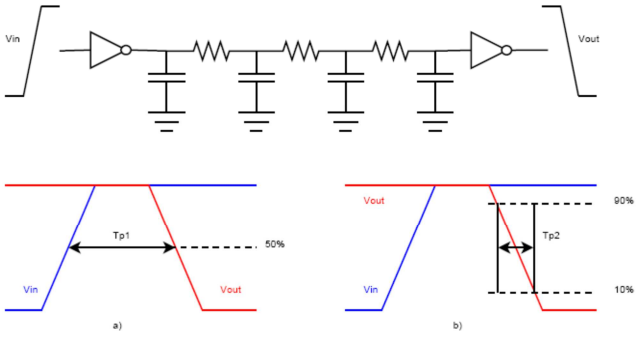


Figure 18: Définitions du temps de propagation sur une interconnexion

Il est possible en utilisant le schéma de la figure 19 d'obtenir une première approximation du temps de propagation au sein d'une interconnexion :

$$T1 = (R_{buffer} + R_{fil})(C_{IN} + C_{fil})$$

- R_{buffer} représente la résistance équivalente de l'inverseur ;
- R_{fil} représente la résistance du fil ;
- C_{IN} représente la capacité d'entrée de l'inverseur en sortie de ligne ;
- C_{fil} représente la capacité totale présentée par le fil.

Tous ces paramètres ont été définis dans la section précédente. Cette première approximation n'utilise pas un modèle distribué pour l'interconnexion mais un modèle lumped ce qui fait qu'elle n'est pas très fiable.

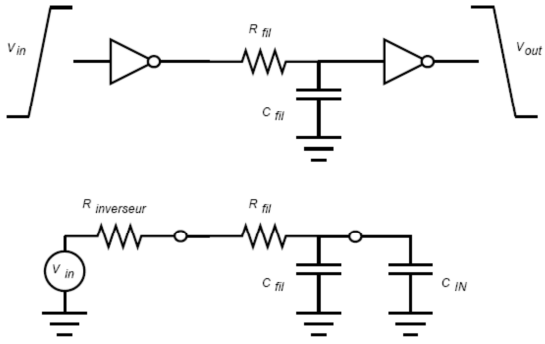


Figure 19: Schéma équivalent lumped d'une interconnexion avec ses buffers d'entrée et de sortie

La formule la plus utilisée pour le calcul du temps de propagation a été introduite par Elmore dans [8]. Bien que cette formule ne considère toujours pas la notion de modèle distribuée des interconnexions, c'est celle que les concepteurs utilisent afin d'obtenir une première estimation rapide et approximative du temps de propagation.

$$T2 = 0.5R_{fil}C_{fil} + R_{buffer}C_{fil} + R_{fil}C_{IN} + R_{buffer}C_{IN} + R_{fil}C_{fil}$$

On peut trouver dans [9] une évolution de la formule présentée par [10] qui prend en compte l'utilisation de modèles distribués pour les lignes d'interconnexion.

$$T3 = 0.1R_{fil}C_{fil} + ln\left(\frac{1}{1-v}\right)(R_{buffer}C_{fil} + R_{fil}C_{IN} + R_{buffer}C_{IN} + 0.4R_{fil}C_{fil})$$

T3 représente le temps de propagation mis pour que la sortie atteigne une valeur normalisée (ici dans le cas de la définition première du temps de propagation on prendra pour valeur finale $V_{dd}/2$) avec v = valeur/2 V_{dd} soit v = 0.5 ce qui donne en remplaçant :

$$T3 = 0.377R_{fil}C_{fil} + 0.693\,(R_{buffer}C_{fil} + R_{fil}C_{IN} + R_{buffer}C_{IN})$$

Méthodes d'insertion de buffer

Avec l'augmentation de la longueur des interconnexions, ainsi qu'avec l'évolution des paramètres technologiques, le temps de propagation au sein des interconnexions devient critique et par conséquent, il faut avoir recours à des méthodes d'insertion de buffers afin d'accélérer la propagation des signaux. Beaucoup de travaux ont étés mené autour de l'insertion de buffers dans les interconnexions [11], [12],[13].

L'expression du premier ordre du temps de propagation sur un fil peut être définie de la manière suivante :

$$T_{fil} = R_{fil/m}C_{fil/m}L^2$$

- $R_{fil/m}$ représente la résistance du fil par unité de longueur ;
- $C_{fil/m}$ représente la capacité du fil par unité de longueur ;
- L représente la longueur du fil.

Le temps de propagation au sein d'une interconnexion évolue en fonction du carré de la longueur du fil. Lorsque l'on insère des buffers, l'interconnexion est fractionnée en plusieurs segments de longueur L/(K−1) où K représente le nombre de buffers insérés le long de la ligne tel que le montre la figure 20 b). L'insertion de buffers permet de réduire la dépendance du temps de propagation par rapport à la longueur du fil de manière quadratique à linéaire. Le fait d'insérer des buffers va ici faire augmenter le temps de propagation dû aux portes puisqu'elles sont plus nombreuses que dans le cas d'une interconnexion non bufferisée. C'est pour cela que les méthodes d'insertion de buffers définissent un nombre optimal de buffers Kopt à insérer ainsi qu'une taille optimale de ces mêmes buffers Hopt afin d'obtenir le temps de propagation minimal sur le fil et donc les performances temporelles maximales.

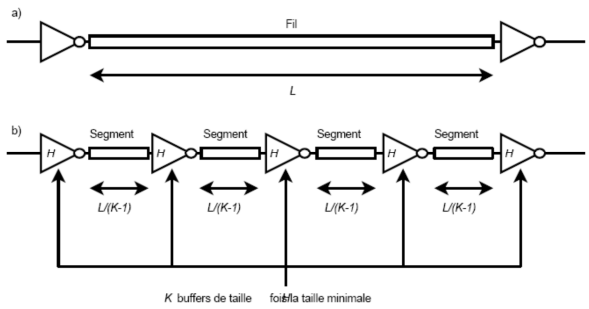


Figure 20: a) Interconnexion classiquement bufferisée; b) Interconnexion bufferisée complètement

3.5 Modélisation de la consommation

L'objectif de la modélisation de la consommation des interconnexions est ici d'obtenir un modèle mathématique dont les paramètres d'entrée seront choisis par l'utilisateur et de fournir des résultats rapides et précis (puisque les expérimentations ont été effectuées au niveau physique) en termes de consommation et de délai. Le second objectif de la modélisation de la consommation est de permettre ensuite de tester rapidement l'efficacité des techniques d'optimisation des performances que l'on rencontre dans la littérature et également de tester celles que nous allons proposer. Afin d'effectuer les expérimentations, il est indispensable d'identifier les différents paramètres qui vont faire évoluer la consommation et de les classer par niveau d'abstraction.

Puisque nous avons choisi d'effectuer la modélisation au niveau physique, le premier paramètre qui rentre en compte dans la modélisation est le choix de la technologie ; ici les expérimentations ont été effectuées pour trois technologies, 130,90 et 65nm. Chaque technologie dispose d'un nombre de couches de métal qui lui est propre ; un second paramètre sera le niveau de métal sur lequel sera placé le bus. Dans chaque technologie on retrouve plusieurs niveaux de métal qui ont chacun une finalité différente et des propriétés physiques (dimensions) différentes :

- La couche de métal la plus basse, de section très faible, sera réservée pour de courtes interconnexions au sein même des cellules (couche locale) ;
- La couche de métal immédiatement supérieure sera quant à elle réservée pour les bus locaux à l'intérieur de ces cellules (couche intermédiaire) ;
- Les couches de métal supérieures seront réservées pour les bus entre les cellules ; ce sont ces couches de métaux qui vont nous intéresser plus particulièrement car c'est sur celles-ci que seront placés les bus (couche intermédiaire) ;
- Les deux dernières couches de métal, les plus hautes, qui ont la section la plus importante seront réservées à la distribution de l'arbre d'horloge ainsi qu'aux lignes d'alimentation (couche globale).

	130nm	90nm	65nm
couche locale	1	1	1
couches intermédiaires	3	4	5
couches globales	2	2	2

Tableau 4: Répartition du nombre et du type de couches de métal en fonction de la technologie

Le tableau 4 présente pour les trois technologies qui ont été modélisées la répartition des couches de métal. Il a été expliqué précédemment que la capacité présentée par le fil va varier en fonction de la couche de métal choisie ; en effet les dimensions (épaisseur, hauteur, largeur, espacement) diffèrent en fonction de la couche sur laquelle se trouve le bus donc les capacités parasites (fil par rapport au substrat, diaphonie crosstalk) qui interviennent dans l'équation de la consommation varient également.

La section précédente a expliqué que lorsqu'une ligne d'interconnexion devient longue on est alors confronté à un problème de temps de propagation, problème que l'on peut résoudre par l'insertion de buffers ; on peut ici en conclure que la longueur de l'interconnexion ainsi que l'insertion de buffers seront des paramètres pour les expérimentations.

Dans la section sur le crosstalk, nous avons vu que selon le type de transition (tableau 3) la capacité de crosstalk pouvait avoir une influence plus ou moins importante sur le retard de propagation en fonction du type de transition observée sur le fil victime. Cette modification de la capacité de crosstalk va également interagir sur la consommation. Chaque type de transition du tableau 4 sera un paramètre pour les expérimentations. Pour résumer, chaque paramètre, sa plage de variation ainsi que son type sont représentés dans le tableau 5. Les différentes expérimentations ont été effectuées avec un simulateur SPICE (ELDO v5.7).

Paramètre	Type	Plage de variation
Technologie	Technologique	130nm / 90nm / 65nm
Couche de métal	Technologique	6 (130nm) / 7 (90nm) / 8 (65nm)
Longueur	Architectural	$0 < L \leqslant 10mm$
Bufferisation	Technologique	Simple / Complète
Transition	Algorithmique	Combinaison

Tableau 5: Paramètres entrant en jeu dans la modélisation de la consommation

Au terme de ces expérimentations, un outil a été développé. Celui-ci fournit à l'utilisateur plusieurs résultats selon la configuration du bus qu'il aura choisi (figure 21). L'utilisateur doit renseigner l'outil sur :

- La technologie ;
- La couche de métal ;
- La longueur du bus ;
- La largeur du bus ;
- La fréquence de fonctionnement ;
- Le type de bufferisation ;
- Le fichier de stimuli (i.e. : les données qui vont circuler sur le bus).

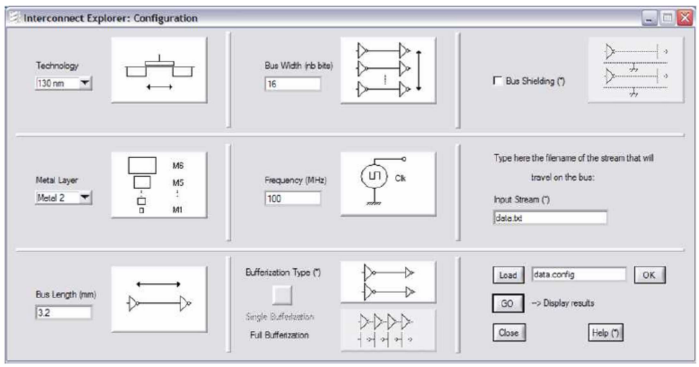


Figure 21: Interface d'entrée de l'outil

L'outil lui fournit en retour des informations sur (figure 22) :

- La consommation énergétique ;
- La consommation de puissance statique ;
- La consommation de puissance moyenne ;
- La consommation de puissance maximale ;

- La fréquence de fonctionnement maximale (déterminée par les transitions pire cas en temporel) ;
- La surface occupée sur la puce (lignes d'interconnexion plus buffers) ;
- Le taux de commutation par bit sur le bus (taux qui sera utile par la suite pour tester les techniques d'optimisation des performances) ;
- Le pourcentage d'apparition de chaque classe de transition du tableau 4 (même remarque que pour le taux de commutation).

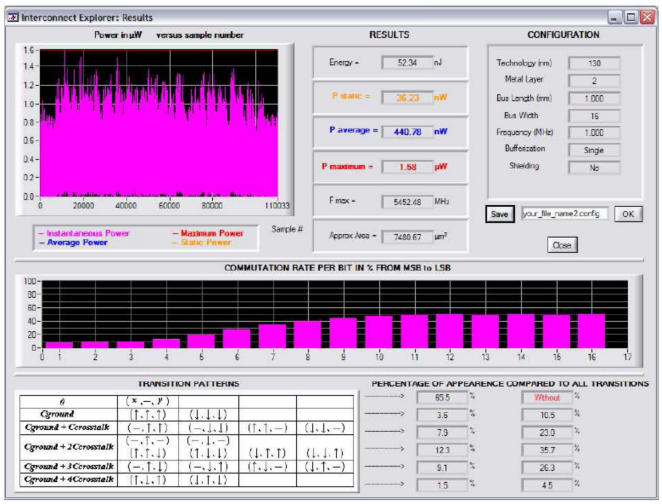


Figure 22: Interface de sortie de l'outil

Des expérimentations ont été effectuées afin de déterminer l'erreur d'estimation entre notre outil et l'outil SPICE. Suite à ces tests, nous avons obtenu une erreur de 3.1% sur un flot de données de type image et 1.2% sur un flot de données de type aléatoire.

3.6 Méthode d'optimisation

Beaucoup de techniques usuelles visent à réduire l'effet du crosstalk dans le but d'accélérer la propagation des informations et également de réduire la consommation des interconnexions. Ces techniques interviennent à différents niveaux d'abstractions [3] mais ont un point commun qui est l'hypothèse que le classement des transitions d'un point de vue temporel sera identique d'un point de vue consommation.

Nous avons donc, dans un premier temps, effectué des expériences afin de vérifier la validité de cette hypothèse et nous nous sommes rendus compte que cette hypothèse était erronée. Nous avons donc proposé une nouvelle classification des transitions d'un point de vue énergétique (cf. tableau 6)

Classification Temporelle		
$(-,-,-)$	0	0 ps
$(\uparrow,\uparrow,\uparrow)$	C_s	49 ps
$(\downarrow,\downarrow,\downarrow)$	C_s	49 ps
$(-,\uparrow,\uparrow)$	C_s+C_c	67 ps
$(-,\downarrow,\downarrow)$	C_s+C_c	67 ps
$(\uparrow,\uparrow,\downarrow)$	C_s+2C_c	99 ps
$(-,\uparrow,-)$	C_s+2C_c	99 ps
$(-,\downarrow,-)$	C_s+2C_c	99 ps
$(\downarrow,\downarrow,\uparrow)$	C_s+2C_c	99 ps
$(-,\uparrow,\downarrow)$	C_s+3C_c	139 ps
$(-,\downarrow,\uparrow)$	C_s+3C_c	139 ps
$(\downarrow,\uparrow,\downarrow)$	C_s+4C_c	173 ps
$(\uparrow,\downarrow,\uparrow)$	C_s+4C_c	173 ps

Classification Energétique		
$(-,-,-)$	0	0.21 fJ
$(\uparrow,\uparrow,\uparrow)$	C_s	13.29 fJ
$(-,\uparrow,\uparrow)$	C_s+C_c	13.43 fJ
$(-,\uparrow,-)$	C_s+2C_c	13.45 fJ
$(\uparrow,\uparrow,\downarrow)$	C_s+2C_c	13.89 fJ
$(-,\uparrow,\downarrow)$	C_s+3C_c	14.10 fJ
$(\downarrow,\uparrow,\downarrow)$	C_s+4C_c	14.86 fJ
$(\downarrow,\downarrow,\downarrow)$	C_s	33.77 fJ
$(-,\downarrow,\downarrow)$	C_s+C_c	92.00 fJ
$(-,\downarrow,-)$	C_s+2C_c	150.35 fJ
$(\downarrow,\downarrow,\uparrow)$	C_s+2C_c	150.73 fJ
$(-,\downarrow,\uparrow)$	C_s+3C_c	207.76 fJ
$(\uparrow,\downarrow,\uparrow)$	C_s+4C_c	265.07 fJ

Tableau 6: Classification des transitions d'un point de vue délai et consommation énergétique

Le tableau 6 montre dans sa partie gauche le temps de propagation et dans sa partie droite la consommation pour les différents types de transitions. Le point saillant de ces résultats est de remarquer que la classification des transitions d'un point de vue consommation n'est pas similaire à celle du point de vue temporel.

La classification en consommation du tableau 6 peut se diviser en deux parties :

- dans la partie haute du tableau, les transitions sont exclusivement montantes et sont classées de celle présentant la plus faible capacité à celle présentant la plus forte ;
- dans la partie basse du tableau, les transitions sont exclusivement descendantes et sont classées de la même manière.

Dans le cas de transitions montantes, la consommation varie seulement de 5.6% autour de la valeur moyenne, elles peuvent donc être toutes classées dans la même catégorie. En fait, la consommation des transitions montantes est due au chemin de court-circuit entre l'alimentation et la masse durant la transition. Durant une transition descendante (donc montante en sortie du premier in verseur), le courant extrait de l'alimentation pour charger la capacité présentée par le fil dépend du type de transition et donc de la capacité du fil. Cette énergie accumulée est ensuite restituée dans la masse pendant le changement d'état lors de la prochaine transition.

3.6.1 Où se situe la consommation ?

La consommation sur les bus est essentiellement de la consommation dynamique qui dépend entre autre de la capacité présentée par la ligne, comme il a été vu précédemment, mais également de l'activité des données sur cette ligne. La figure 23 présente l'activité de chaque fil sur un bus, la part de la somme de consommation pour plusieurs bits de poids faible par rapport à la consommation totale ainsi que le pourcentage d'apparition de chaque classe de transition pour un flot de données de type image et un bus de 8 bits.

Sur cette figure, la consommation (pour un flot de type données) se situe essentiellement sur les bits de poids faible (déjà plus de 50% de la consommation pour les trois derniers bits). Par conséquent, appliquer les techniques d'optimisation sur tout le bus (et notamment sur les MSB) aura un effet très amoindri puisque l'activité des MSB est faible.

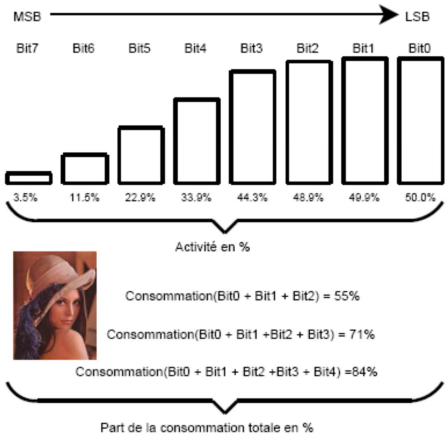


Figure 23: Activité de chaque fil sur un bus 8 bits pour un flot de type image

De plus, un autre résultat intéressant du tableau 7 est que les techniques ayant pour but de supprimer les pires cas (i.e. 1+4r et/ou 1+3r) ne suppriment qu'une partie infime de la totalité des transitions (seulement 1% pour celles qui éliminent les transitions de type 1+4r par exemple). La suppression de ces transitions va certes diminuer le temps de propagation sur le bus, mais ne va pas forcément diminuer la consommation. En effet, ce type de transition sera remplacé par d'autres transitions dont la consommation peut éventuellement être plus importante (si on transforme une transition montante en une transition descendante par exemple).

Facteur de délai g du fil victime	Pourcentage d'appartition de la classe de transition
$g = 0$	66.9%
$g = 1$	2.4%
$g = 1 + r$	6.2%
$g = 1 + 2r$	12.7%
$g = 1 + 3r$	10.8%
$g = 1 + 4r$	1.0%

Tableau 7: Pourcentage d'apparition de chacune des classes de transitions pour un flot de données de type image sur un bus de 8 bits

3.6.2 Méthode d'optimisation proposée

En se basant sur les différentes études faites, nous avons pu mettre en évidence les points principaux à prendre en compte afin de proposer une méthode d'optimisation performante. Ces points principaux sont :

- ne pas se focaliser uniquement sur les transitions de type 1+3r et 1+4r parce que elles ne sont pas forcément les plus significatives dans le nombre total des transitions ;
- travailler là où l'activité est la plus importante sur le bus (ie : les LSB) car ce sont les lignes les plus consommatrices ;
- éviter le plus possible les transitions descendantes. Un point clé pour l'optimisation est de coder les données de telle manière que les transitions descendantes sur le bus apparaissent avec la plus faible capacité possible pour le fil victime et donc consomme le moins d'énergie ;
- avoir un surcoût matériel des codecs le plus faible possible impliquant donc des techniques simples.

La méthode que nous avons proposée est basée sur une architecture de codage classique consistant en un codeur à l'entrée du bus et en un décodeur à la sortie ; cette technique a pour nom le Spatial Switching (technique brevetée [14]). Le codeur a pour but de transformer les n bits de la donnée en entrée en m bits de données sur le bus codé (avec n ≤ m) ; le décodeur réalise l'opération inverse afin de récupérer les données originales. Quand le Spatial Switching est utilisé, le bus est divisé en groupes de deux fils. Chaque paire de fil est codée en utilisant le principe décrit ci-après. Pour chaque paire de fils, un fil de contrôle supplémentaire (noté Switch) est ajouté en sortie du bloc de codage tel qu'illustré sur la figure 24. Par conséquent, pour un bus de taille n, un total de $\frac{n}{2}$ fils supplémentaires est nécessaire.

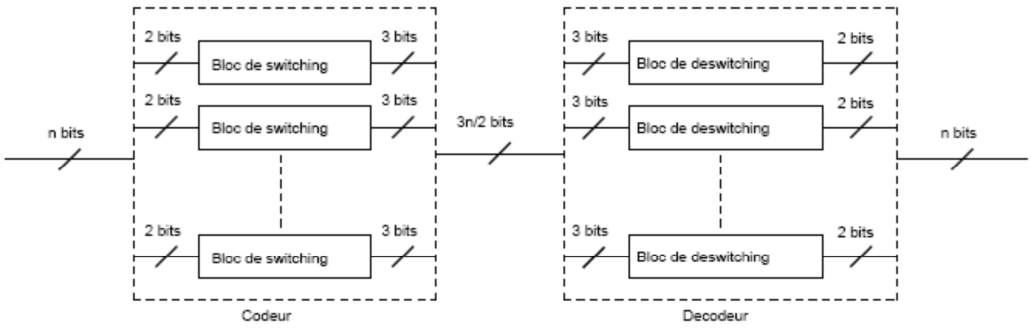


Figure 24: Architectures macro-blocs des codeurs et décodeurs utilisés pour le spatial switching

Le principe du codage est de remplacer les transitions les plus consommatrices sur le bus par d'autres moins consommatrices de telle sorte que le bus présente la plus faible capacité de crosstalk possible. Le point clé de la technique est de détecter toutes les transitions croisées sur des fils adjacents. Ainsi, éliminer les transitions croisées (une transition croisée contient forcément une transition descendante) permet de supprimer les transitions les plus consommatrices telles que référencées dans le tableau 6. Eliminer ces transitions permet également une accélération de la propagation des données sur le bus car il ne reste plus qu'au pire cas des transitions de type 1+2r sur les fils codés.

De plus, comme nous l'avons vu le fait d'éliminer les transitions croisées permet de diminuer le bruit généré sur les interconnexions. Par conséquent, la technique du Spatial Switching permet également de rendre la transmission plus fiable en diminuant potentiellement le taux d'erreur binaire sur les lignes par la suppression de ce type de transitions. Dans le cas où une transition croisée est détectée sur deux fils adjacents, les chemins empruntés par les données sont alors inversés (i.e. soit deux fils 1 et 2 ; si une transition croisée est détectée, la donnée qui devait initialement transiter sur le fil 1, transitera sur le fil 2. Il en va de même pour la donnée devant initialement transiter sur le fil 2). Un signal est également positionné sur le fil de contrôle afin de permettre au décodeur de rediriger correctement les données en sortie du bus. Un exemple pédagogique illustrant le fonctionnement du Spatial Switching est présenté sur le la figure 25. Le flot de données original contient deux transitions croisées. En utilisant le Spatial Switching, le flot de données codées ne contient plus aucune transition croisée.

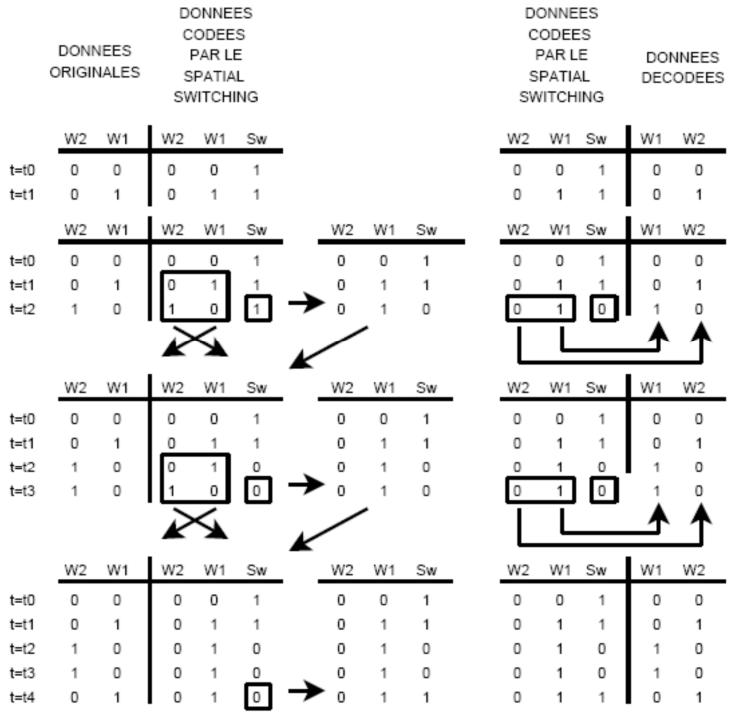


Figure 25: Exemple du Spatial Switching appliqué sur un flot de données, où W_i représente le $i^{ème}$ fil et S_w le signal de contrôle Switch

La figure 26 présente également le bloc de routage des données sur les deux fils voisins. Ce bloc consiste en deux multiplexeurs 2 vers 1 dont les sorties dépendent évidemment du niveau logique du signal de contrôle Switch. En fin de bus, un autre bloc de routage est utilisé (également piloté par le signal de contrôle Switch) pour rediriger les données codées afin de récupérer les données originales sur les deux fils voisins.

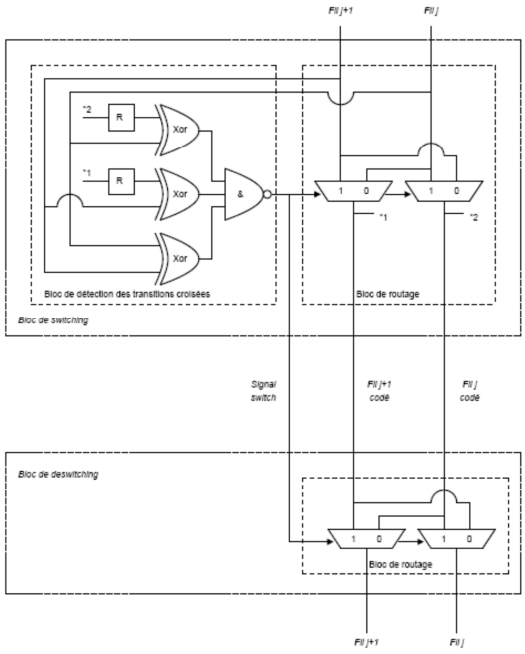


Figure 26: Architecture interne des macro-blocs de switching et bloc de deswitching des codecs présentés à la figure 24

Le tableau 8 présente les résultats obtenus pour différentes technologies (de 130 à 65nm), pour différents niveau de métallisation (Layer 3, 5, 6 et 7) et pour différentes longueurs d'interconnexion (3 et 5 mm) lorsque des données de type image sont propagées sur le bus.

	130nm / Layer 3 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	2.78	4.06	5.20	2.64
6λ	1.09	0.68	0.13	-4.12
8λ	-0.52	-2.55	-4.70	-10.56
10λ	-2.11	-5.74	-9.49	-16.95
12λ	-3.75	-9.02	-14.41	-23.51

	90nm / Layer 3 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	4.42	7.52	10.44	10.00
6λ	3.11	4.88	6.49	4.73
8λ	2.09	2.86	3.45	0.68
10λ	1.85	2.38	2.73	-0.28
12λ	0.41	-0.52	-1.61	-6.07

	65nm / Layer 3 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	4.06	7.11	10.19	10.20
6λ	3.63	6.25	8.90	8.48
8λ	3.18	5.35	7.54	6.67
10λ	2.71	4.41	6.13	4.79
12λ	2.20	3.38	4.60	2.74

	130nm / Layer 5 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	3.66	5.77	7.54	5.80
6λ	2.26	2.96	3.32	0.18
8λ	0.94	0.33	-0.62	-5.08
10λ	-0.31	-2.18	-4.39	-10.09
12λ	-1.58	-4.71	-8.19	-15.16

	90nm / Layer 6 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	4.39	7.74	10.36	9.95
6λ	3.18	5.05	6.73	5.10
8λ	2.21	3.11	3.82	1.22
10λ	1.30	1.30	1.10	-2.40
12λ	0.54	-0.24	-1.20	-5.47

	65nm / Layer 7 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	4.15	7.26	10.46	10.50
6λ	3.74	6.44	9.22	8.86
8λ	3.31	5.57	7.92	7.12
10λ	2.84	4.64	6.53	5.26
12λ	2.37	3.70	5.12	3.38

	130nm / Layer 5 / 5mm			
	2LSB	4LSB	6LSB	8LSB
4λ	4.84	8.19	11.19	10.60
6λ	4.04	6.59	8.79	7.40
8λ	3.14	4.79	6.10	3.81
10λ	2.23	2.97	3.37	0.17
12λ	1.36	1.22	0.74	-3.33

	90nm / Layer 6 / 5mm			
	2LSB	4LSB	6LSB	8LSB
4λ	4.95	8.35	11.75	12.05
6λ	4.31	7.06	9.82	9.47
8λ	3.78	6.01	8.25	7.38
10λ	3.22	4.89	6.56	5.13
12λ	2.60	3.66	4.71	2.66

	65nm / Layer 7 / 5mm			
	2LSB	4LSB	6LSB	8LSB
4λ	4.11	7.38	10.64	10.55
6λ	3.87	6.90	9.92	9.59
8λ	3.62	6.40	9.18	8.60
10λ	3.36	5.88	8.40	7.57
12λ	3.10	5.38	7.61	6.51

Tableau 8: Variation de la consommation sur le bus

Les valeurs données dans le tableau sont des gains en % et inclues bien sûr la consommation due aux codecs que nous rajoutons. Lorsqu'un gain est négatif (cases grisées) ceci signifie que notre solution est plus coûteuse que de ne rien rajouter ; ceci étant dû au fait que la consommation des codecs dépasse la réduction en consommation réalisée sur le bus par notre solution.

Premièrement, l'analyse des résultats permet de remarquer que l'efficacité du Spatial Switching augmente lors des sauts technologiques, ce qui est un point très important pour la réduction dela consommation dans les technologies actuelles et futures.

Deuxièmement, nous pouvons remarquer que le Spatial Switching est même efficace pour les couches de métal les plus basses et devient de plus en plus efficace avec l'élévation dans les couches de métal ; c'est sur les couches hautes, réservées plus particulièrement pour les bus que le gain sera encore plus important.

Troisièmement, les résultats montrent que plus le bus est long, meilleure est la réduction de consommation. Le gain en consommation (pour les longueurs simulées) peut atteindre 12% pour un bus de 5mm, ce gain augmente encore si la longueur augmente. Dans les SoC actuels, les bus peuvent varier en moyenne de 1 à 7mm pour des bus très longs. Par conséquent, pour cette plage de variation de longueur, le Spatial Switching peut apporter une réduction de consommation conséquente.

Quatrièmement, comme souligné dans le chapitre précédent, il existe une valeur optimale du nombre de bits sur lequel la technique peut être appliquée. Le Spatial Switching a pour but d'éliminer les transitions croisées, or, ce type de transition a une probabilité d'apparition plus forte

sur les fils dont l'activité est importante (i.e. les LSB). Les résultats du tableau 8 montrent que la technique permet d'obtenir un accroissement des gains en consommation lorsqu'elle est appliquée sur les 2, 4 et 6 LSB. Les gains commencent à décroître pour les 8 LSB. Ceci est dû au fait que les MSB sont plus corrélés entre eux que les LSB ; il y a donc moins de transitions croisées sur ces premiers. Par conséquent, les codecs placés sur les 2 MSB consomment plus que la réduction de consommation qu'ils apportent. Cette valeur optimale du nombre de bits peut être différente en fonction du type de données qui circulent sur le bus. Ici les résultats présentés dans le tableau 8 sont obtenus pour le transfert d'un flot de données de type image. Or dans le cadre d'autres applications du TDSI (génération des données en sortie d'un papillon de FFT, entrelacement temporel des données . . .) il se peut que le profil d'activité des données devienne totalement aléatoire. Les résultats du tableau 9 montrent alors que pour ce type de profil d'activité, il est intéressant d'appliquer le Spatial Switching sur la totalité du bus. De plus amples détails sur cette méthode d'optimisation sont présentés dans l'article fourni en Annexe C

	130nm / Layer 3 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	1.75	2.99	4.23	5.72
6λ	0.60	0.68	0.77	1.12
8λ	-0.50	-1.51	-2.52	-3.27
10λ	-1.59	-3.68	-5.78	-7.72
12λ	-2.71	-5.92	-9.13	-12.09

	90nm / Layer 3 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	2.76	5.06	7.52	10.13
6λ	1.88	3.31	4.88	6.61
8λ	1.21	1.96	2.86	3.92
10λ	1.05	1.64	2.38	3.28
12λ	0.08	-0.29	-0.51	-0.58

	65nm / Layer 3 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	2.57	4.82	7.06	9.47
6λ	2.28	4.24	6.19	8.31
8λ	1.97	3.62	5.27	7.09
10λ	1.66	2.99	4.32	5.82
12λ	1.31	2.30	3.29	4.44

	130nm / Layer 5 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	2.74	4.70	6.86	9.21
6λ	1.79	2.81	4.02	5.42
8λ	0.91	1.04	1.37	1.89
10λ	0.07	-0.64	-1.16	-1.48
12λ	-0.78	-2.35	-3.72	-4.89

	90nm / Layer 6 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	2.76	5.23	7.56	10.18
6λ	1.95	3.60	5.12	6.92
8λ	1.30	2.30	3.16	4.41
10λ	0.69	1.08	1.33	1.88
12λ	0.17	0.05	-0.21	-0.18

	65nm / Layer 7 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	2.62	4.78	7.09	9.56
6λ	2.35	4.23	6.26	8.45
8λ	2.06	3.64	5.38	7.28
10λ	1.75	3.02	4.45	6.04
12λ	1.43	2.39	3.51	4.78

	130nm / Layer 5 / 5mm			
	2LSB	4LSB	6LSB	8LSB
4λ	3.26	5.92	8.69	11.71
6λ	2.73	4.86	7.10	9.59
8λ	2.13	3.66	5.31	7.21
10λ	1.53	2.46	3.50	4.79
12λ	0.95	1.29	1.76	2.47

	90nm / Layer 6 / 5mm			
	2LSB	4LSB	6LSB	8LSB
4λ	3.14	5.75	8.45	11.33
6λ	2.71	4.88	7.15	9.59
8λ	2.36	4.18	6.09	8.18
10λ	1.98	3.43	4.96	6.67
12λ	1.56	2.59	3.71	5.01

	65nm / Layer 7 / 5mm			
	2LSB	4LSB	6LSB	8LSB
4λ	2.70	5.12	7.64	10.34
6λ	2.54	4.80	7.16	9.70
8λ	2.38	4.47	6.66	9.04
10λ	2.21	4.13	7.15	8.35
12λ	2.03	3.78	5.62	7.65

Tableau 9: Variation de la consommation sur un bus pour un flot de données aléatoire

3.7 Conclusion & Réflexion

Ce chapitre a abordé la problématique de l'estimation et l'optimisation de la consommation des interconnexions. Nous avons vu qu'il n'était pas trivial de réaliser un modèle de consommation d'une interconnexion mais que cette étape était capitale si nous voulions pouvoir proposer des méthodes d'optimisation. En effet, la modélisation ne peut se faire qu'au niveau physique afin d'obtenir un modèle précis et fiable. De plus, dû à la réduction des dimensions technologiques, nous sommes obligés de prendre en compte l'effet de couplage capacitif qui peut survenir lorsque les lignes d'interconnexions sont parallèles sur une distance importante (plus la technologie est agressive et plus cette distance se réduit). Nous avons vu que l'effet de crosstalk avait deux effets principaux qui sont la modification du délai de transmission de la donnée sur la ligne et l'augmentation de la consommation du transport de cette information due aux capacités de couplage.

Un outil d'estimation a été développé afin de permettre aux concepteurs d'analyser la consommation de leur réseau d'interconnexions en fonction du flux de données qui y transite. Dans un second temps, nous avons proposé une nouvelle classification des transitions d'un point de vue consommation ; classification qui n'existait pas jusque-là. Cette classification était indispensable afin de pouvoir proposer des méthodes d'optimisation de la consommation cohérentes. En effet, jusqu'alors, les différentes méthodes proposées s'appuyaient sur la classification des transitions d'un point de vue délai or, nous avons montré que cette classification n'était pas valide d'un point de vue consommation. A partir de nos résultats, nous avons proposé une nouvelle approche permettant d'optimiser la consommation du réseau d'interconnexions en limitant le nombre de transitions croisées qui sont les plus pénalisantes d'un point de vue consommation. Nous avons également montré qu'en fonction du flux de données utilisé, il n'était pas judicieux d'appliquer notre méthode dite du Spatial Switching sur l'ensemble du bus mais qu'il était préférable de se focaliser sur les bits de poids faibles (pour les données de type image par exemple car les pixels sont très corrélés entre eux). Nous avons également vu que notre approche ne générait pas toujours un gain dès lors que nous prenions en considération la consommation des codecs ajoutés dans le système de communication. Grâce à l'outil Interconnect Explorer, nous pouvons rapidement voir si l'optimisation est viable ou non sans devoir repasser par des phases de simulation SPICE.

Jusqu'à présent ces travaux se sont focalisés sur les interconnexions point à point dans les SoC ; ces connexions sont certes existantes mais dans les puces actuelles, massivement parallèles (MPSoC), elles ne sont plus suffisantes pour faire transiter tout le flux d'information. Les concepteurs ont aujourd'hui besoin d'intégrer des moyens de communications plus efficaces que du point à point ou que du bus partagés et donc ont recours à des réseaux sur puce (Network on Chip). Une nouvelle thèse, celle d'Erwan Moréac, est en cours afin de voir comment tirer profit de notre expérience sur les interconnexions point à point dans la problématique de l'optimisation de la consommation et de la latence dans les NoC.

L'idée première est d'étudier la modélisation en consommation des NoC qui est aujourd'hui réalisée, de l'améliorer au besoin et de proposer de nouvelles pistes d'optimisation. Les premiers travaux que nous avons menés montrent que les routeurs des NoC sont bien modélisés via des simulations bas niveaux (SPICE) sur des technologies ASICs mais que les liens de communications entre les routeurs sont, eux, grossièrement modélisé (pas de prise en compte du crosstalk par exemple). L'idée est donc de proposer un nouveau modèle pour ces liens en se basant sur les travaux d'Antoine Courtay et en les mettant à jour avec les technologies actuelles. Nous avons également besoin d'intégrer les modèles de consommation dans des simulateurs plus haut niveau afin de limiter le temps de simulation. Nous avons choisi d'utiliser le simulateur Noxim qui utilise le modèle de consommation DSENT (modèle le plus usuel dans la littérature aujourd'hui), de le modifier afin de pouvoir le rendre bit près (il est déjà cycle près). En effet, si nous voulons bien prendre en compte le crosstalk cela nécessite de connaître à chaque temps de cycle les données qui transitent sur les liens entre les routeurs. Une fois cette modification réalisée, nous mettrons à jour les modèles de consommation des liens, avec prise en compte du crosstalk, sur des technologies plus actuelles (<45nm). L'impact du crosstalk augmentant avec la diminution de la consommation i est indispensable de réaliser ces travaux avant toutes propositions d'optimisations.

Dans un second temps, nous appliquerons la méthode du Spatial Switching afin d'étudier, si cette technique développée pour les interconnexions point à point, peut s'avérer pertinente pour le

contexte des réseaux sur puce. Enfin, nous essaierons de proposer de nouvelles méthodes d'optimisation afin de réduire conjointement le délai et la consommation sur le réseau. Ces deux facteurs pouvant, selon le cas, être orthogonaux il faudra trouver des méthodes permettant de réaliser un compromis qui pourra être pondéré en fonction du poids de chacun des paramètres (délai/consommation) dans les performances attendues pour le système complet.

Références

[1] W.J. DALLY AND J.W. POULTON. DIGITAL SYSTEMS ENGINEERING. CAMBRIDGE UNIVERSITY PRESS, 1998.

[2] J.M. RABAEY, A. CHANDRAKASAN, AND B. NIKOLIC. DIGITAL INTEGRATED CIRCUITS : A DESIGN PERSPECTIVE. PEARSON EDUCATION, 2003.

[3] A. COURTAY. CONSOMMATION D'ENERGIE DANS LES INTERCONNEXIONS SUR PUCE: ESTIMATION DE HAUT NIVEAU ET OPTIMISATIONS ARCHITECTURALES, THESE DE DOCTORAT, 2008.

[4] ITRS. TECHNICAL REPORT. INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS, 2002.

[5] ITRS. TECHNICAL REPORT. INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS, 2004.

[6] ITRS. TECHNICAL REPORT. INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS, 2006.

[7] R. HO, K. MAI, AND M. HOROWITZ. THE FUTURE OF WIRES. PROCEEDINGS . IEEE, 89(4) :490–504, APRIL 2001.

[8] PTM. INTERCONNECT. PREDICTIVE TECHNOLOGY MODEL, 2007.

[9] W.C. ELMORE. THE TRANSIENT RESPONSE OF DAMPED LINEAR NETWORKS WITH PARTICULAR REGARD TO WIDEBAND AMPLIFIERS. JOURNAL OF APPLIED PHYSICS, VOL 19 :55–63, JAN 1948.

[10] T. SAKURAI. CLOSED-FORM EXPRESSIONS FOR INTERCONNECTION DELAY, COUPLING, AND CROSSTALK IN VLSI'S. IN IEEE TRANS ON ELECTRON. DEVICES, VOLUME VOL 40, PAGES 118–124, 1993.

[11] H.B. BAKAGLU AND J.D. MEINDL. OPTIMAL INTERCONNECTION CIRCUITS FOR VLSI. IEEE TRANS ON ELECTRON. DEVICES, VOL 32(NO.5) :903–909, 1985.

[12] G. CHEN AND E.G. FRIEDMAN. LOW-POWER REPEATERS DRIVING RC AND RLC INTERCONNECTS WITH DELAY AND BANDWIDTH CONSTRAINTS. IEEE TRANS. ON VERY LARGE SCALE INTEGRATION SYSTEMS, VOL 14(NO.2) :161–172, 2006.

[13] X.C. LI, J.F. M, H.F. HUANG, AND Y.LIU. GLOBAL INTERCONNECT WIDTH AND SPACING OPTIMIZATION FOR LATENCY, BANDWIDTH AND POWER DISSIPATION. IEEE TRANS ON ELECTRON. DEVICES, VOL 52(NO.10) :2272–2279, 2005.

[14] A. COURTAY, J. LAURENT, O. SENTIEYS, AND N. JULIEN. PROCEDE ET DISPOSITIF DE CODAGE, SYSTEME ELECTRONIQUE ET SUPPORT D'ENREGISTREMENT ASSOCIES. PATENT PENDING, 2008. BFF 08P0103/HC.

Chapitre 4

Conception de systèmes pervasifs pour le domaine maritime

Résumé :

Ce dernier chapitre aborde le dernier thème de recherche sur lequel je travaille à savoir la conception de systèmes pervasifs pour le domaine maritime. Nous présentons dans ce chapitre comment la mesure de performance, et particulièrement la mesure du vent réel, est un enjeu primordial dans le monde de la course à la voile. Dans un premier temps nous expliquons quels sont les différents verrous à lever afin de mesurer le vent réel sur un bateau à voile. Dans un second temps nous présentons nos apports en termes de modélisation et discutons des résultats obtenus. Enfin ce chapitre se conclut en discutant des autres travaux réalisés ou en cours dans cette thématique et les futurs développements.

4.1 Introduction

Ce qui caractérise un système pervasif est le fait qu'il est généralement enfoui dans son environnement et qu'il a à ou doit communiquer avec son environnement. Cette communication peut se faire soit pour renvoyer des données à un système tiers soit pour lui permettre de s'adapter à son environnement. L'adaptation du système est un paramètre incontournable dans le domaine maritime car la mer est par définition un environnement incertain générant de fortes contraintes qui peuvent modifier le comportement intrinsèque du système. Pour bien comprendre le concept, je vais prendre un exemple pédagogique mais réaliste.

Supposons un voilier de croisière en plein océan dont l'énergie électrique n'est fournie que par des énergies renouvelables (solaire, éolien et hydraulique) ; ce cas de figure fait l'objet du projet AMI Voilier du Futur. Tous les systèmes électriques du bateau, que ce soit les systèmes de navigation/sécurité (pilote, AIS, radar…) ou les systèmes de confort (climatisation, douche, multimédia…) sont donc dépendants de ces 3 producteurs. Supposons maintenant qu'un de ces systèmes ait une panne (par exemple l'hydrogénérateur), la production électrique va donc chuter fortement. Or si les navigants ne s'en rendent pas compte rapidement la réserve d'énergie des batteries va elle aussi chuter mettant potentiellement en péril l'équipage. Si un système pervasif s'occupe de la gestion des énergies à bord, celui va s'apercevoir de la baisse de production, via ses capteurs, et va ré estimer les besoins et la production énergétique envisageable sur la route suivie. Suite à cette analyse, complètement autonome (d'où le nom de pervasif), il va alors prendre des décisions par exemple de délestage, de modification de la qualité de service pour certains appareils etc… afin d'optimiser les réserves. Il peut également proposer une nouvelle route aux utilisateurs afin de maximiser la production d'énergie avec les producteurs restants et bien sûr informer les navigants de la panne.

Après avoir donné ma définition d'un système pervasif, nous allons aborder les travaux de recherche autour de cet axe. Les premiers travaux datent de 2010 et ont été réalisés dans le cadre de la thèse Cifre de Ronan Douguet ; thèse au sein du Groupama Sailing Team de Franck Cammas. La problématique était la mesure de performance des voiliers de compétitions ; ceux sont ces travaux qui seront développés dans ce chapitre. Durant cette thèse, nous avons lancé de nouveaux travaux autour de la réalité augmentée pour l'aide à la navigation : thèse de Jean Christophe Morgère (ARED région Bretagne + CG56). L'idée de ces travaux était de fournir aux plaisanciers des informations triées et pertinentes afin qu'ils puissent naviguer le plus surement possible ; ces travaux font l'objet d'une maturation via la SATT Ouest Valorisation. Suite à la reconnaissance de nos compétences dans le domaine des systèmes embarqués pour le domaine maritime, nous avons été contactés afin de

rentrer dans le consortium d'un projet dans le cadre de l'AMI Navire du Futur. L'idée du projet Voilier du Futur est de concevoir un voilier ayant l'empreinte écologique la plus faible possible par l'utilisation de matériaux bio-sourcés et par l'utilisation exclusive d'énergies renouvelables afin de produire l'électricité du bord. Notre travail dans ce projet est de développer un gestionnaire d'énergie temps réel et autonome afin de garantir une qualité de service acceptable pour les navigants ; ces travaux « font l'objet » de la thèse de Mathilde Tréhin (le projet est arrêté aujourd'hui pour des problèmes de financement). Enfin depuis un an, nous avons une thèse Cifre qui a démarré avec la société NKE Marine sur l'amélioration de la chaîne de mesure de vent afin de développer un nouveau pilote automatique ; ces travaux se font dans le cadre de la thèse d'Hugo Kerhascouët.

Les premiers et les derniers travaux sur cette thématique ont attrait à la mesure des performances du bateau et donc attrait à la mesure du vent qui est la force vélique pour les voiliers. A première vue, mesurer le vent parait simple puisque des capteurs de vent existent ; en fait cette mesure est loin d'être triviale et cela fait maintenant plus de 30 ans que des gens y travaillent.

4.2 Mesure des performances du bateau

4.2.1 Mesure du vent

Un skipper sur un bateau a besoin de connaitre la force du vent réel (True Wind Speed) et l'angle du vent réel (True Wind Angle) afin d'analyser les performances de son bateau. Cependant lorsque que le bateau navigue, le capteur de vent mesure la vitesse du vent apparent (Apparent Wind Speed) et l'angle du vent apparent (Apparent Wind Angle). Afin de calculer l'angle et la vitesse du vent réel nous avons donc besoin d'une autre donnée qui est la vitesse du bateau (Boat SPeed) comme le montre la figure 27.

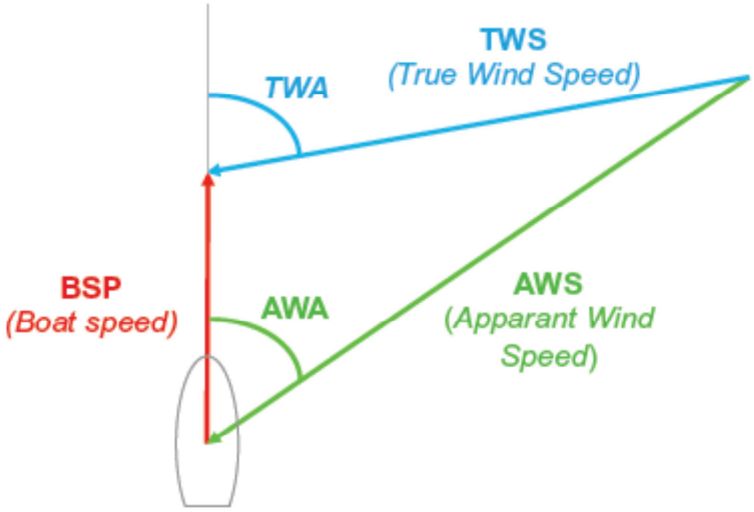


Figure 27: Schéma simplifié de la mesure du vent

Le calcul du vent réel pourrait être relativement simple d'après la figure ci-dessus si les mesures des 2 capteurs (vent & vitesse) n'étaient pas perturbées par d'autres phénomènes. La mesure du vent (angle & vitesse) se fait en utilisant un anémomètre-girouette (aussi appelé anémo-girouette) à coupelles comme présenté sur la figure 28.

Figure 28: Anémomètre-girouette à coupelles (B&G)

L'anémomètre est composé de 3 demi-coquilles identiques disposées sur des bras horizontaux écartés de 120° les uns des autres. Un aimant situé sur l'axe de rotation permet de fournir un signal dont la période correspond à un tour des coupelles. La girouette est quant à elle constituée d'une pale équipée d'un aimant sur son axe vertical. Cet aimant vient exciter deux ou trois capteurs à effet hall positionnés respectivement autour de l'axe de la girouette à 90° ou 120° les uns des autres (cela dépend du constructeur). Ce type de capteur est généralement monté en tête de mât du bateau car c'est l'endroit le « moins perturbé » du bateau comme le montre la figure 29.

Figure 29: Positionnement du capteur de vent

4.2.2 Mesure de la vitesse du bateau

La mesure de la vitesse du voilier est le deuxième paramètre clé pour déterminer correctement le vent réel. Ce paramètre peut être mesuré avec plusieurs instruments et n'a pas tout le temps la même signification. En effet, le déplacement du voilier peut se décomposer en deux routes : le déplacement fond et le déplacement surface (cf. figure 30). Cette section permet d'expliquer chaque déplacement et de présenter les capteurs de vitesse qui leur sont associés.

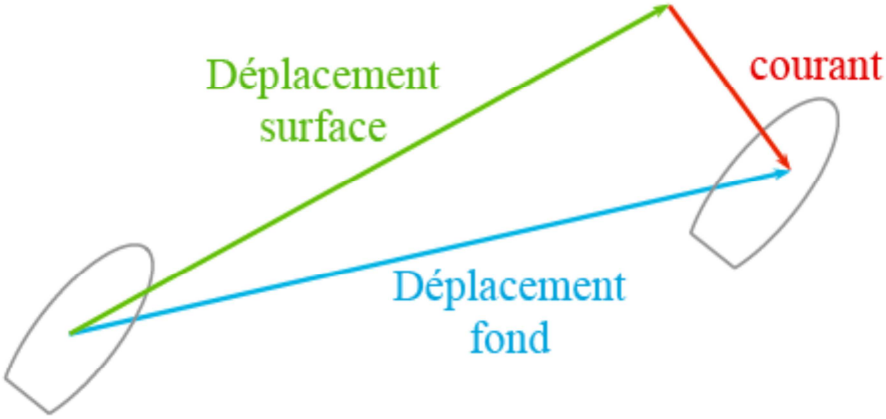


Figure 30: Déplacement surface et déplacement fond

La vitesse surface : La vitesse surface correspond à la vitesse de déplacement du voilier par rapport à la masse d'eau où il navigue. Cette vitesse est indépendante du courant et est mesurée généralement par une roue à aubes située sur le devant du bulbe de quille.

La vitesse fond : La vitesse fond correspond à la vitesse de déplacement du voilier par rapport à la terre. Elle correspond au déplacement surface couplé au déplacement de la mer par rapport à la terre (cf. figure 30). Cette vitesse est donc dépendante du courant et elle est généralement mesurée par le GPS.

4.2.3 Les perturbations de mesure

Comme nous l'avons dit précédemment, les mesures de ces captures subissent un certain nombre de perturbation que nous allons expliquer plus avant.

L'alignement de la girouette : La première perturbation intervient sur la mesure de l'angle du vent lors du montage du capteur. En effet, il est difficile d'aligner correctement la girouette sur l'axe du bateau. Cette erreur doit être identifiée, sinon l'angle de vent apparent sera différent selon que l'on navigue tribord amure ou bâbord amure. Afin d'évaluer cet offset (cf. figure 31), le navigateur procédera à une phase de calibration pour obtenir une mesure symétrique du vent apparent sur les deux bords.

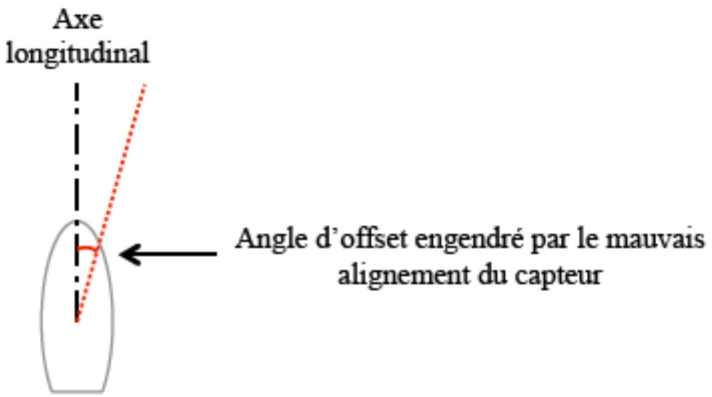


Figure 31: Décalage de la girouette

Torsion du mât : Les mâts importants, comme par exemple ceux des bateaux du Vendée Globe (supérieurs à 30 m), subissent une torsion de quelques degrés (cf. figure 32) qui engendre une erreur sur la mesure de l'angle du vent. Cet angle nommé angle de « twist » peut varier de +/- 5° sur ce type de bateau.

Figure 32: Angle de torsion

Les mouvements du voilier : Les mouvements du voilier, le roulis, le tangage et le lacet créent une composante de vent en tête de mât qui perturbe sa mesure. Les mouvements du bateau sont présentés sur la figure 33.

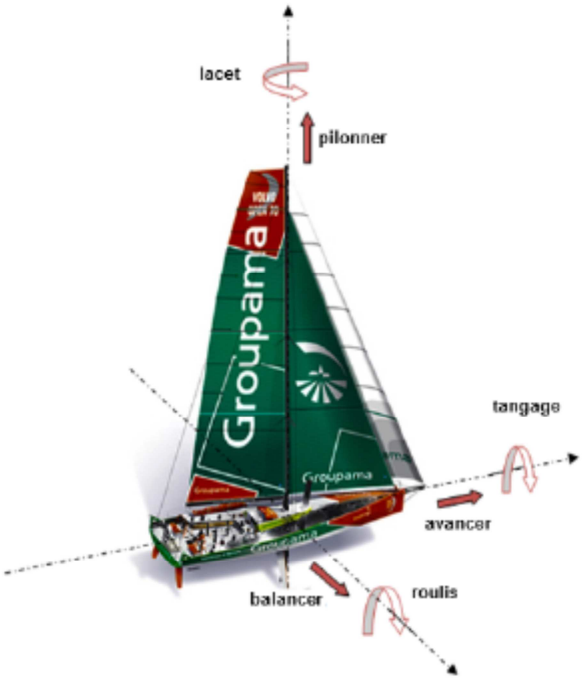


Figure 33: Mouvements du bateau

À partir des vitesses angulaires mesurées par l'IMU (Inertial Motion Unit) située en pied de mât et de la hauteur de celui-ci (correspondant au bras de levier), la vitesse tangentielle, engendrée par chaque mouvement en tête de mât, peut être déterminée. En additionnant les trois vitesses tangentielles dues au roulis, au tangage et au lacet, il est possible de calculer le vecteur de vent apparent créé par les mouvements du bateau.

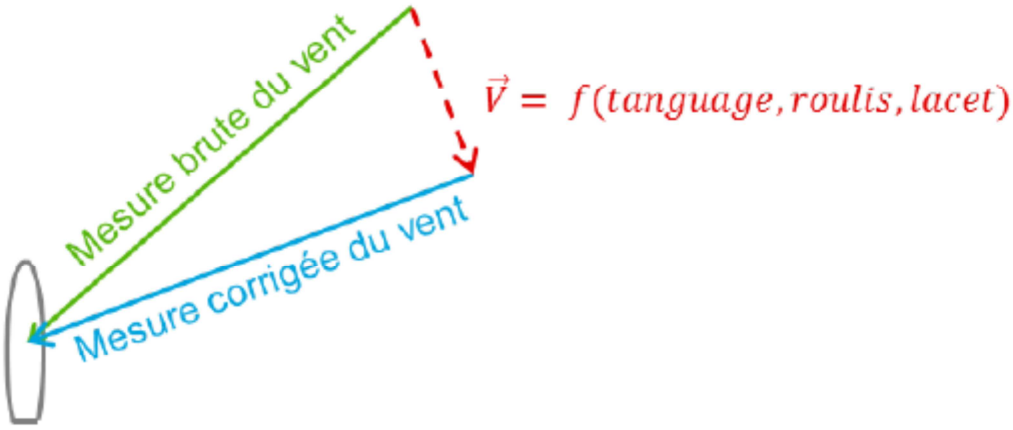


Figure 34: Correction des mouvements du bateau

Cette erreur est ensuite soustraite aux mesures brutes du vent pour retrouver un vent apparent corrigé (cf. figure 34). Cette correction repose sur l'hypothèse que la plateforme est rigide et donc que les mesures de l'IMU sont corrélées avec les mesures du vent [1].

La dérive du bateau : La dérive du bateau est l'un des paramètres clés pour l'analyse des performances et pour le calcul du vent réel. La décomposition des déplacements du voilier montre qu'il ne dérive pas seulement avec le courant, mais aussi avec le vent. En effet, lorsqu'un voilier navigue il est soumis à différentes forces, notamment la force vélique. Cette dernière peut se diviser en deux composantes : la force propulsive et la force de dérive (cf. figure 35, [2]). La force propulsive fait avancer le bateau sur son axe longitudinal alors que la force de dérive fait déraper le bateau sur son axe transversal. Cela entraîne un angle de dérive qui est nommé angle de « leeway ».

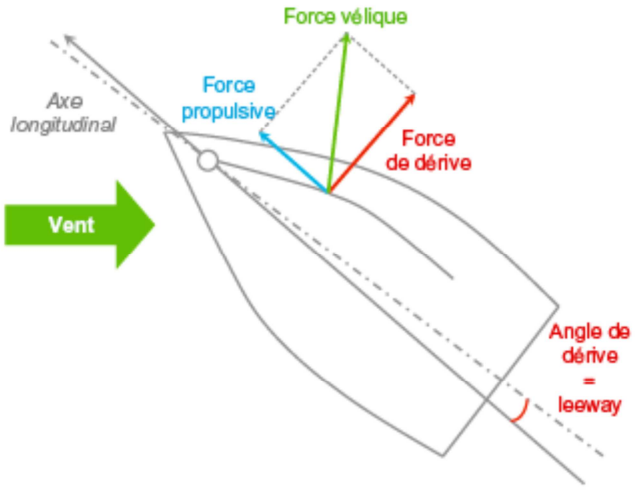


Figure 35: Schéma de la décomposition de la force vélique

Cet angle de leeway induit une erreur dans la mesure de l'angle du vent apparent qui est ensuite répercutée sur la mesure du vent réel. En effet, l'angle de vent apparent mesuré ne prend pas en compte la dérive du voilier. Cependant pour analyser les performances de son voilier, le navigateur doit, par exemple, connaître son angle de remonté au vent en tenant compte de la dérive du voilier due au vent. La figure 36 montre la perturbation du leeway sur le calcul du vent réel.

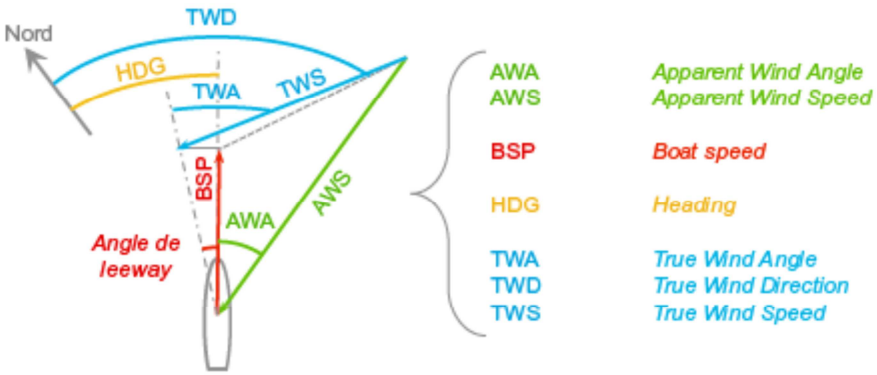


Figure 36: Schéma du calcul du vent réel

Actuellement, pour corriger cette erreur, les centrales de navigation utilisent une formule empirique, définie en 1981 [3], permettant de calculer l'angle de leeway. Cette formule (ci-dessous) dépend de la vitesse du voilier (BSP), de l'angle de gîte (Heel) ainsi que d'une constante k.

$$leeway = k \times \frac{Heel}{BSP^2}$$

Cette équation ayant été définie pour un monocoque de la coupe de l'America ne peut être appliquée pour les multicoques. De plus, très peu de paramètres sont pris en compte dans cette formule. Elle décrit simplement un fait : plus le voilier gîte plus il dérive et plus il avance vite moins il dérive. D'autre part, la constante k, pouvant être ajustée entre 9 et 16 [4], est configurée empiriquement en fonction du voilier et des résultats obtenus. Le manque de paramètres et la configuration empirique de cette formule sont tels que celle-ci est rarement exploitable en pratique pour le calcul du vent réel.

L'écoulement du vent : L'écoulement du vent autour du bateau est perturbé par les voiles ; ce phénomène est appelé « upwash ». En analysant l'écoulement du vent sur la voile, nous remarquons que lorsque les flux d'air s'écoulant du coté intrados et extrados de le voile se rejoignent sur le bord de fuite des voiles, ils créent des vortex [5] comme illustré dans la figure 37. Lorsque le voilier navigue, la mesure du vent subie entièrement cette perturbation. Pour diminuer l'erreur liée à ce phénomène, le capteur de vent est surélevé de 1,50 m sur une perche en carbone mais cela ne suffit pas. D'après une étude théorique réalisée pour un bateau de la Volvo Ocean Race, le vent devrait être mesuré plus de 16 m au-dessus de la tête de mât afin de ne plus être perturbé par l'upwash [6], or une mesure du vent à cette hauteur est inconcevable en pratique.

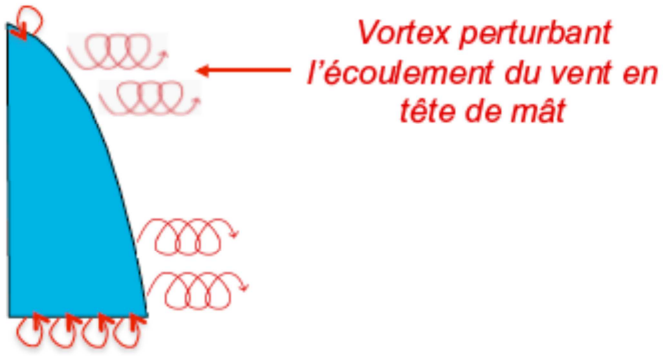


Figure 37: Effet d'upwash

Pour corriger l'upwash, les centrales de navigation utilisent des tables de correction qui dépendent de la vitesse et de l'angle du vent réel. Des heures de navigations sont nécessaires pour configurer ces tables afin d'obtenir des corrections convenables dans toutes les conditions de navigation [7].

Le cisaillement du vent : Le cisaillement du vent, appelé « wind shear », correspond à la variation de la direction et de la vitesse du vent en fonction de l'altitude. Dans le cadre de la voile, cela correspond à la variation de vent entre le pied et la tête de mât. Plusieurs causes peuvent expliquer ce phénomène [8] notamment la différence de température entre l'air et l'eau ou encore la friction du vent avec la surface de la mer qui entraîne une diminution de la vitesse du vent (voir figure 38). Le wind shear n'est pas calculé par les centrales de navigation mais généralement il y a la possibilité de configurer une valeur d'offset sur l'angle et sur la vitesse du vent réel pour compenser ce phénomène. Le problème est que ces valeurs d'offset sont définies empiriquement par le navigateur.

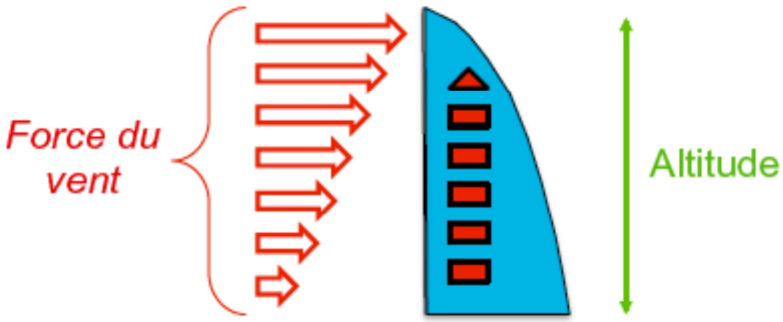


Figure 38: Diminution de la vitesse du vent à l'approche de la surface de la mer

Fluctuation de la masse volumique du vent : La masse volumique de l'air ne perturbe pas la mesure du vent mais influence les performances du voilier. En effet, les navigants ont pu observer que pour une vitesse et un angle de vent réel identique, le voilier ne se comporte pas tout le temps de la même façon. Ce phénomène s'explique par la variation de la masse volumique de l'air puisque celle-ci influence la force exercée sur la voile. Comme la masse volumique de l'air dépend de plusieurs paramètres météorologiques (pression atmosphérique, température, humidité), il est possible d'observer des performances différentes du voilier avec la même vitesse et le même angle du vent réel. Pour compenser ce phénomène, il faudrait déterminer des polaires du bateau dépendantes de la masse volumique de l'air ; aucune centrale de navigation ne propose de solution pour prendre en compte ce phénomène. Ce paramètre n'a pas été pris en compte lors de notre étude.

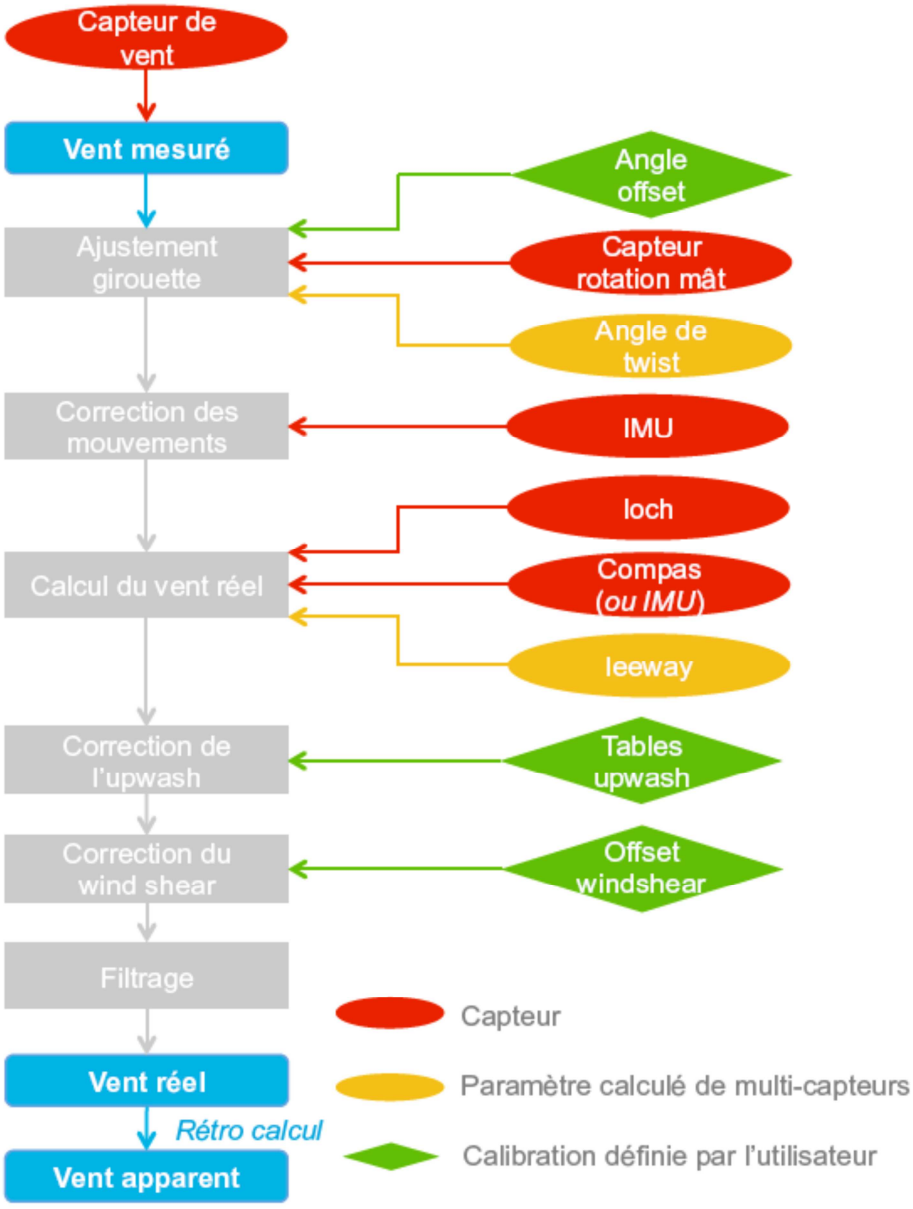


Figure 39: Chaine de correction du vent

Les perturbations décrites dans cette partie montrent toute la complexité de la mesure du vent sur un navire : obtenir une mesure précise du vent nécessite plusieurs étapes de correction. C'est pourquoi, l'utilisation d'un système multi-capteurs est nécessaire pour fusionner les données afin de corriger la mesure du vent. Tous les capteurs sont donc centralisés autour d'une plateforme embarquée, appelée centrale de navigation, qui calibre, corrige et calcule toutes les informations requises pour l'analyse des performances. Malheureusement, malgré tous les capteurs dont nous disposons, des perturbations comme le leeway, l'upwash ou le windshear restent corrigées de manière empirique.

Afin de calculer les valeurs du vent réel, nous utilisons les différentes étapes de corrections présentées dans la figure 39 ; cette procédure est appelée chaîne de correction du vent. Dans la section suivante, nous verrons plus en détail notre apport sur la mesure du vent réel à savoir le calcul du « leeway ».

4.3 Méthodologie de calcul du leeway

L'analyse de la chaîne de mesure du vent a mis en exergue plusieurs défauts dans la mesure de ce paramètre ; une de ces erreurs est le calcul empirique qui permet l'obtention de l'angle du leeway. Nous allons donc nous attacher à mieux évaluer cette valeur en proposant une nouvelle méthodologie. La décomposition de la force vélique en deux éléments permet de comprendre la cause du leeway (cf. figure 40) : une force propulsive fait avancer le voilier sur son axe longitudinal et une force de dérive le fait déraper sur son axe transversal. C'est cet angle de leeway que le navigateur cherche à connaître pour optimiser les réglages de son voilier et déterminer son angle optimal de remontée au vent.

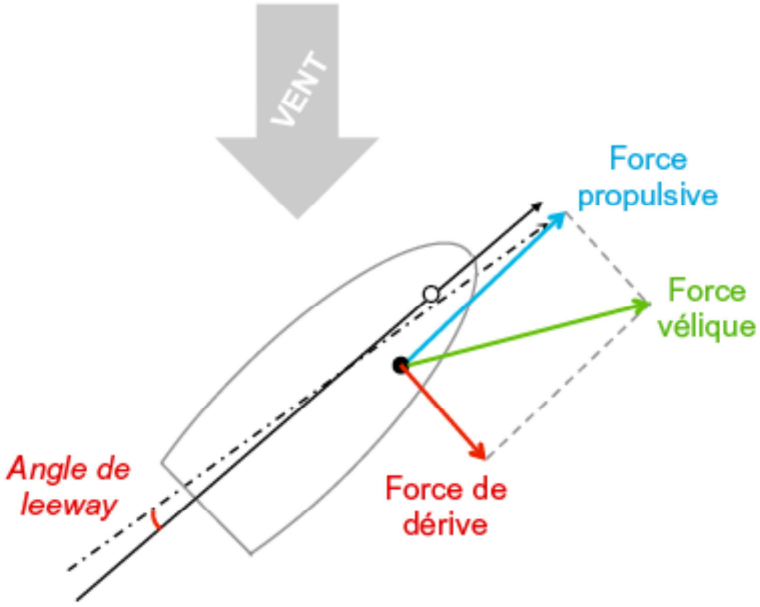


Figure 40: Décomposition de la force vélique

La principale difficulté pour mesurer le leeway réside dans le fait qu'un voilier ne dérive pas seulement à cause du vent mais aussi à cause du courant marin. La dérive globale du voilier est donc composée du leeway et du courant comme le montre la figure 41. Trois déplacements du voilier se distinguent : le déplacement apparent, le déplacement surface et le déplacement fond.

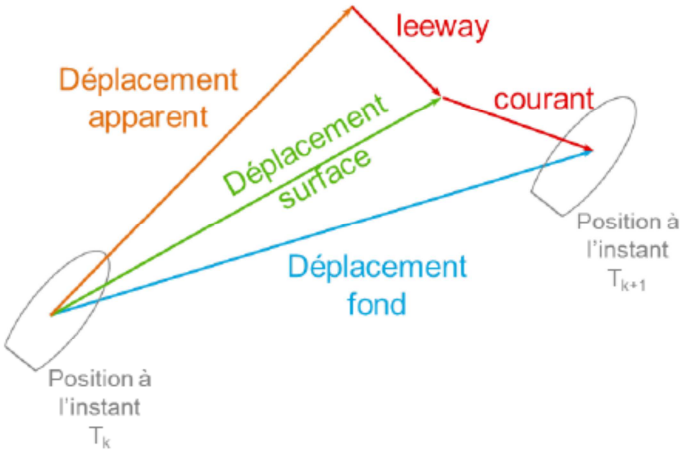


Figure 41: Décomposition du déplacement d'un voilier

La figure 42 détaille les paramètres des différents déplacements : la route apparente est définie par le cap du compas (HDG) et la vitesse du loch (BSP), la route surface est définie par le cap du compas (HDG), l'angle de leeway (LEE) et la vitesse sur cet axe (BSP_LEE) et la route fond est définie par le cap fond (COG) et la vitesse fond (SOG). Avec les capteurs couramment installés sur les voiliers de course (GPS, loch et compas), la dérive globale du voilier peut aisément être calculée. Par contre, il est actuellement impossible de différencier la partie liée au leeway de la partie liée au courant. Pour identifier ces paramètres, il faudrait connaître le vecteur de déplacement surface.

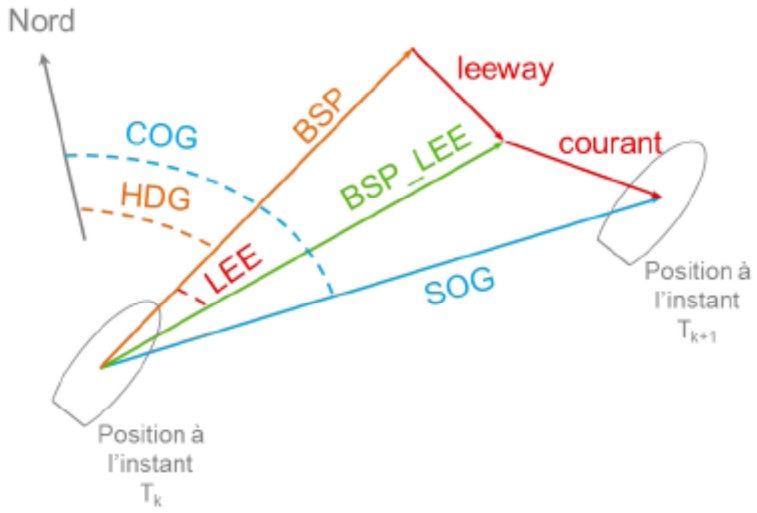


Figure 42: Analyse des déplacements d'un voilier

4.3.1 Modèle global d'estimation du leeway

L'ensemble du système d'estimation du leeway et du courant s'appuie sur deux filtres de Kalman ; le premier filtre permet d'estimer le vecteur courant (V_{xcur},V_{ycur}) dans un repère cartésien et le second estime le vecteur leeway (V_{xlee},V_{ylee}) dans ce même repère. L'hypothèse faite sur ce système est de considérer le courant constant pendant une certaine période de temps. Pour notre cas d'étude, nous nous basons sur une fréquence d'échantillonnage de 10 Hz car c'est la fréquence maximale utilisée dans les centrales de navigation ; celle-ci est jugée suffisante pour calculer correctement le vent et le déplacement du bateau. Les résultats, détaillés plus loin, sont obtenus à partir de 400 points échantillonnés ce qui représente une période de 40 s. En admettant une vitesse du voilier maximale de 40 nœuds, celui-ci parcourra 0.44 mille nautique soit l'équivalent d'environ 815 m sur une période de 40 s. Nous considérons ainsi que la mer est uniforme sur une distance maximale de 815 m, ce qui est vraisemblable par rapport à la réalité. La première étape de notre modèle est d'initialiser le courant à partir de la formule empirique du leeway. Nous émettons ainsi l'hypothèse que l'estimation du vecteur de courant faite après l'initialisation, sera assez proche de la valeur du courant de référence. Ensuite, cette estimation sera prise en compte dans le second filtre afin de calculer le vecteur du leeway. Enfin, après une étape de transformation permettant d'obtenir l'angle du leeway à partir du vecteur, nous pourrons reboucler le système sur lui-même afin d'affiner l'estimation du leeway ; le schéma général de ce système est visible sur la figure 43.

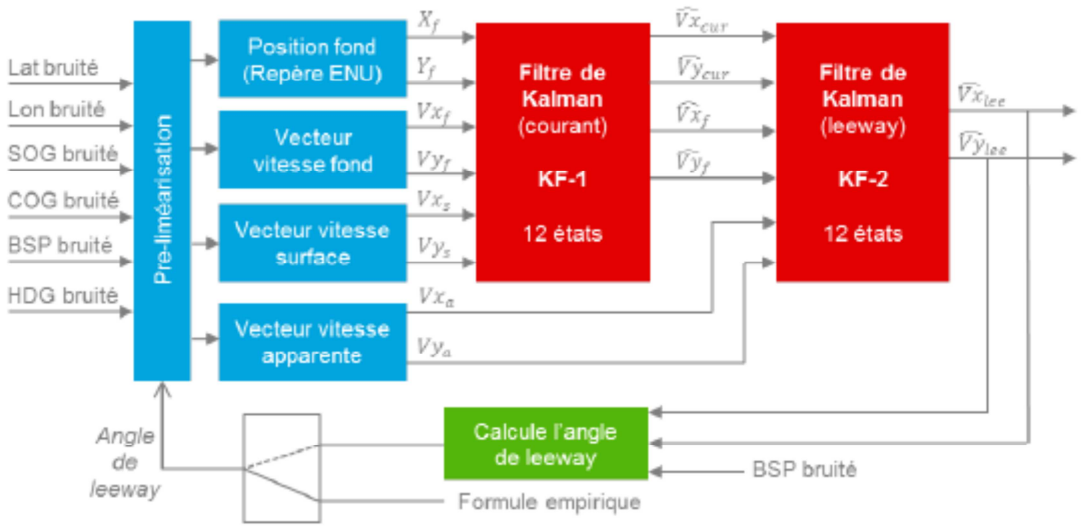


Figure 43: Système d'estimation leeway/courant

L'intérêt de séparer le système en deux filtres est aussi de réduire la complexité de celui-ci. En effet, des travaux [9] montrent l'utilité de décomposer un filtre de Kalman en deux filtres de tailles inférieures de manière à diminuer le nombre d'opérations en virgule flottante. Plus d'information sur cette décomposition sont présenté dans [10]. L'étape de « pré-linéarisation » est essentielle pour appliquer le filtre de Kalman puisque celui-ci ne peut que décrire l'évolution d'un système linéaire. La matrice de transition F de l'équation d'état et la matrice de transition H de l'équation d'observation doivent donc être constantes. Les limites du filtre de Kalman nous ont donc incités à décrire l'évolution des déplacements du bateau et de ces perturbations (leeway et courant) dans un plan cartésien afin de disposer de relations linéaires. Cette étape désignée « pré-linéarisation » correspond en réalité à une transformation permettant d'exprimer toutes les mesures dont nous disposons dans un même repère ; ces mesures sont ainsi exprimées dans un repère ENU (East North Up) dans lequel l'axe du « Up » est inutile pour le problème posé (c.f. [10]).

Le modèle KF-1 permet de prédire le déplacement fond (position, vitesse et accélération), le déplacement surface (vitesse et accélération) ainsi que le courant (vitesse). L'objectif de ce filtre est de minimiser l'erreur liant le déplacement fond et surface au courant

$$\varepsilon = \vec{V}_{fond} - (\vec{V}_{surface} + \vec{V}_{courant})$$

Afin de mettre à jour les prédictions du filtre, les mesures du GPS, du compas, du loch ainsi que le calcul de la formule empirique du leeway sont utilisées à travers l'étape de pré-linéarisation. Les paramètres du vecteur de mesure et du vecteur d'état sont illustrés sur la figure 44.

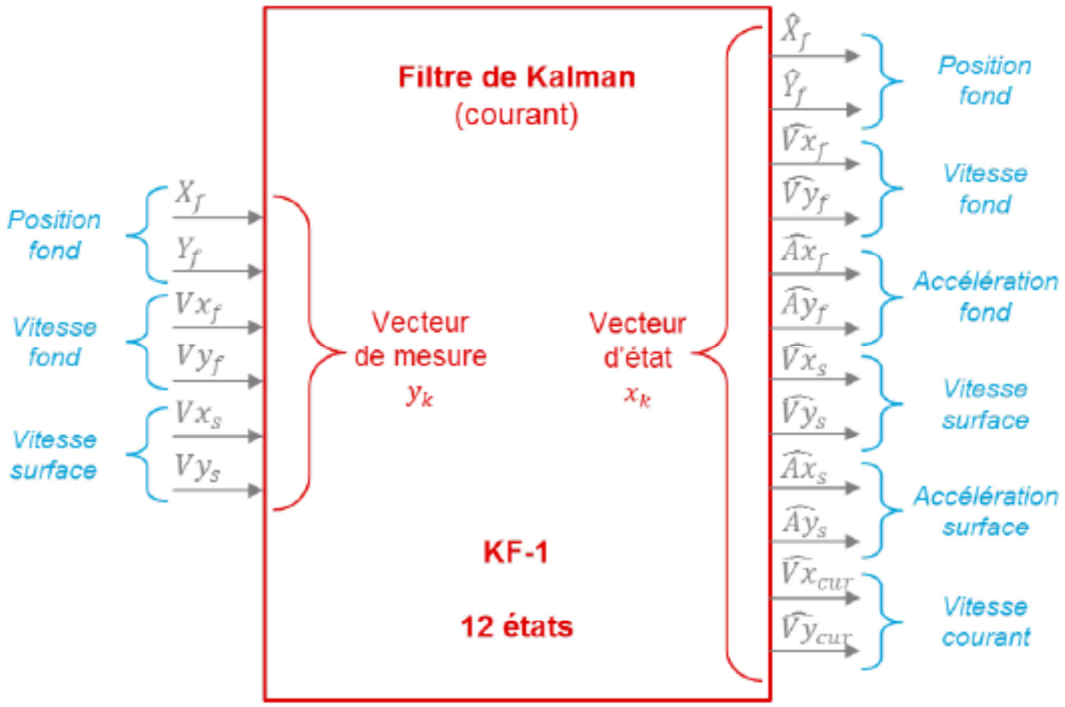


Figure 44: Filtre de Kalman KF-1

Les configurations des différentes matrices du modèle KF-1 sont présentées plus avant dans [10].

Le modèle KF-2 permet de prédire les vitesses et les accélérations concernant le déplacement fond, le déplacement apparent et le leeway. L'objectif de ce filtre est de minimiser l'erreur entre la vitesse fond $\vec{V}_{\text{fond}}$ et l'estimation de la vitesse fond comme étant la somme de la vitesse apparente $\vec{V}_{\text{apparent}}$, de la vitesse du leeway $\vec{V}_{\text{leeway}}$ et de la vitesse du courant $\vec{V}_{\text{courant}}$. Les prédictions du filtre sont ensuite mises à jour avec les paramètres du vecteur de mesure ; la vitesse du courant, la vitesse fond et la vitesse apparente. Les vecteurs d'état et de mesure sont illustrés sur la figure 45. L'estimation du leeway obtenue en sortie du filtre KF-2 est ensuite utilisée en entrée du filtre KF-1 lorsque l'initialisation du système est terminée et que le courant a convergé vers une valeur stable.

4.3.2 Résultats

Dans cette section, nous présenterons les résultats sur 2 cas de figure : le premier lorsque l'erreur sur la formule empirique est faible et le second lorsque une erreur conséquente est simulée. Afin d'analyser les résultats de ce filtre, nous proposons à chaque fois de visualiser la vitesse (CurR) et la direction (CurD) du courant ainsi que l'angle du leeway. Ces paramètres étant calculés d'après les vecteurs de vitesse du courant et de vitesse du leeway. Toutes ces données sont comparées aux données non bruitées utilisées dans notre simulateur [10].

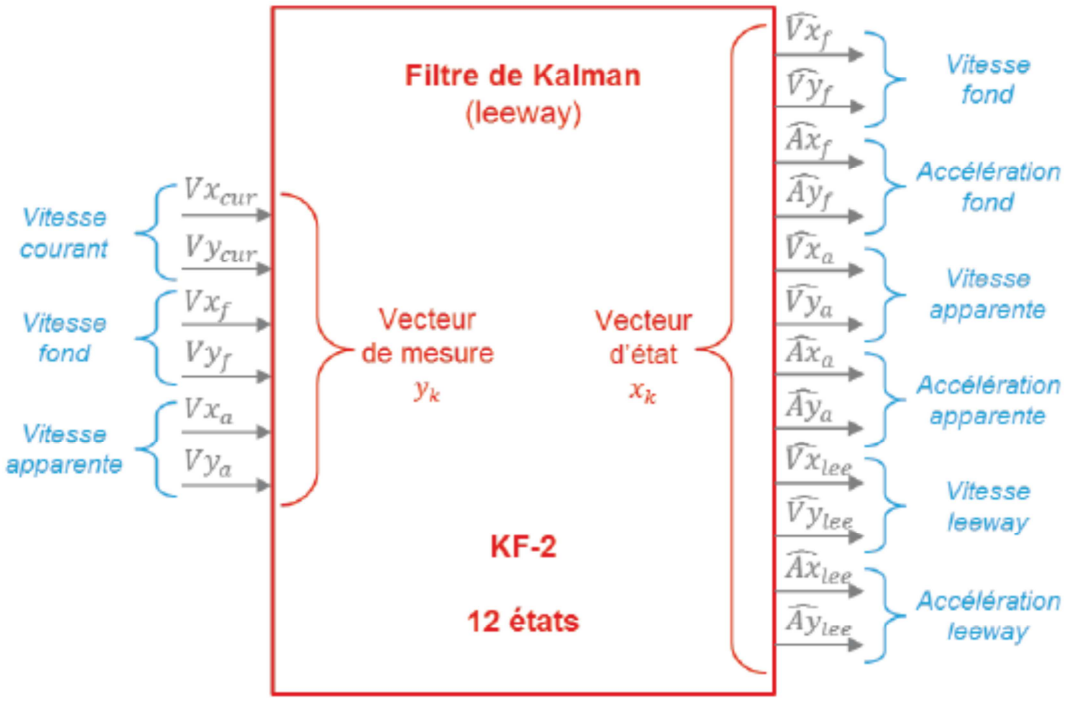


Figure 45: Filtre de Kalman KF-2

Les résultats (cf. figure 46) montrent l'estimation de la vitesse et de la direction du courant ; le bleu représente la valeur de référence et le rouge correspond à l'estimation calculée. Une phase d'initialisation du modèle, utilisant la formule empirique du leeway, est nécessaire pour que le courant se stabilise ; sa durée est fixé aléatoirement à 200 échantillons (20s à 10Hz). Une fois cette phase terminée, le système est rebouclé sur lui-même puisque le leeway calculé en sortie du filtre KF-2 est utilisé en entrée du filtre FK-1.

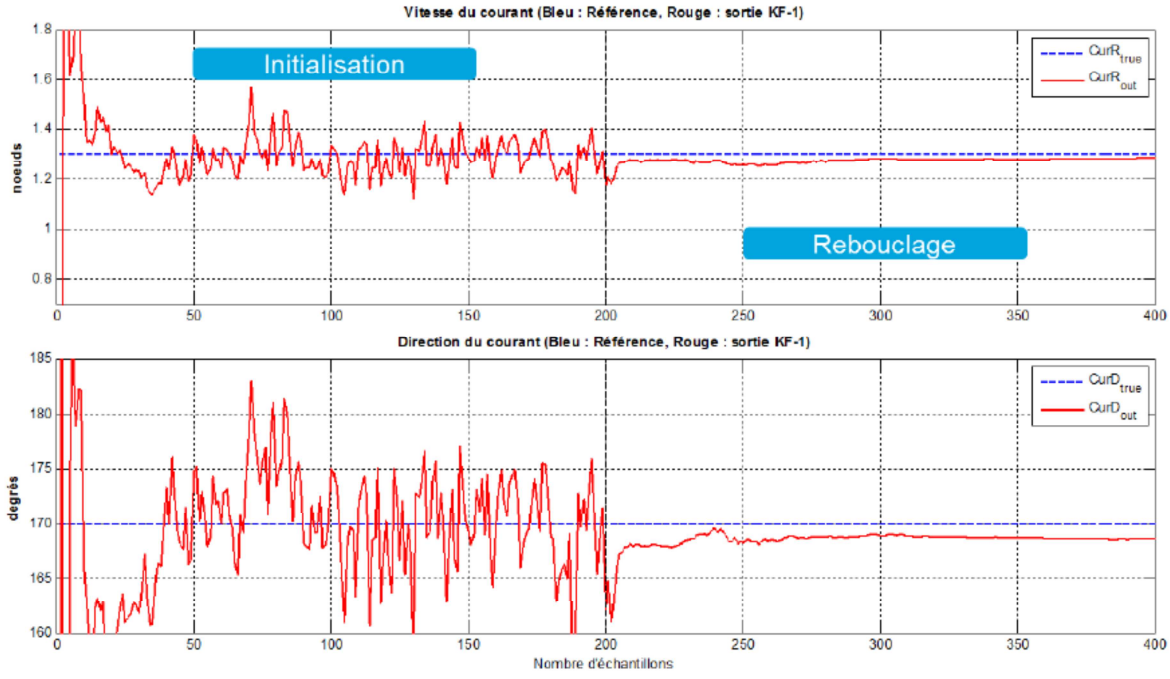


Figure 46: Estimation du courant marin

La figure 47 dévoile les résultats obtenus pour la mesure de l'angle du leeway. Le premier graphe (en mauve) représente la valeur du leeway calculé avec la formule empirique et le second représente la référence (en bleu) et la valeur du leeway estimée par le modèle (en rouge). Le tableau 10 représente l'erreur moyenne obtenue pendant toute la durée du rebouclage (200 échantillons). En considérant que le paramètre k de la formule empirique du leeway n'est pas trop éloigné de la vérité et que cette formule est seulement bruitée par les mesures provenant du loch et de la centrale inertielle, nous obtenons des résultats proches des références malgré une erreur moyenne sur le leeway de 7%.

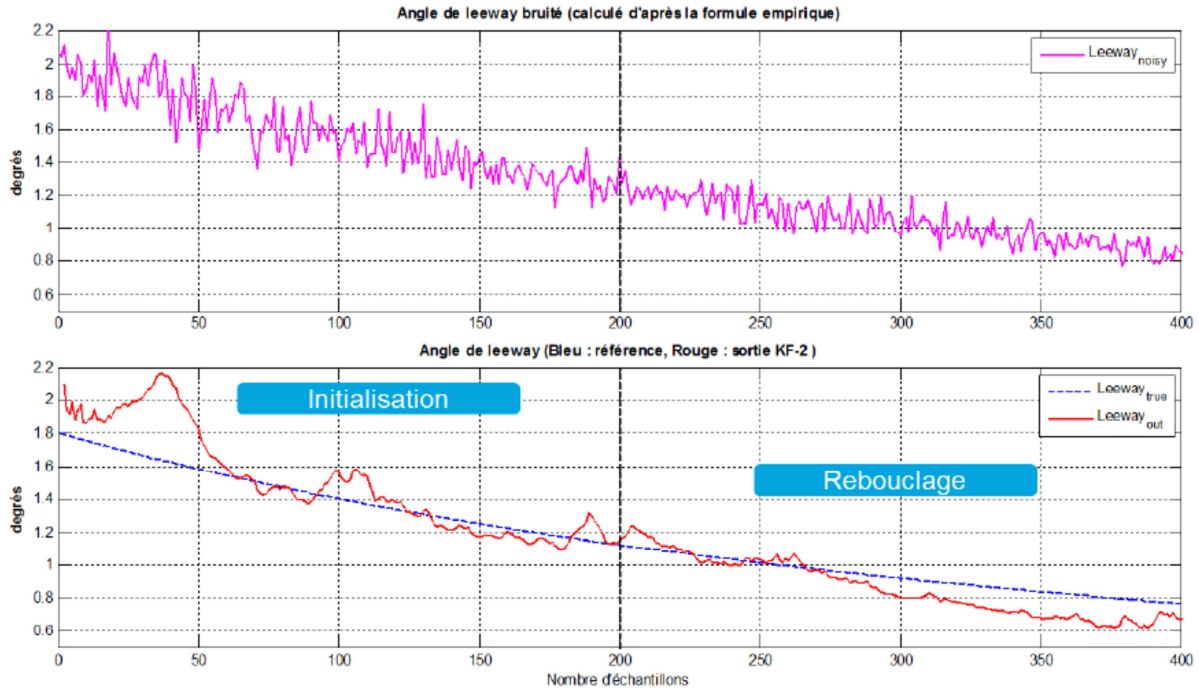


Figure 47: Estimation du leeway (cas N°1)

Paramètres	Unité	Erreur Moy.	Erreur Moy. (%)
CurR	Noeuds	0.024	1.85 %
CurD	Degrés	1.400	0.82 %
Leeway	Degrés	0.065	7.00 %

Tableau 10: Erreur moyenne

La seconde étude permet de vérifier le comportement du filtre lorsque l'erreur sur la formule empirique du leeway est conséquente. C'est pourquoi le simulateur du jeu de données a été initialisé avec une constante k différente de celle utilisée dans le modèle. N'ayant aucune indication sur l'incertitude de la formule empirique du leeway et sur la constante k, nous émettons l'hypothèse d'une erreur de 25 % sur le paramètre.

La figure 48 présente les résultats obtenus pour la vitesse et la direction du courant marin. De la même manière que pour l'étude précédente, le bleu représente la référence et le rouge l'estimation du modèle. D'après la visualisation des courbes dans la phase de rebouclage, les erreurs semblent relativement faibles, néanmoins cela représente 7 % d'erreur sur la vitesse du courant.

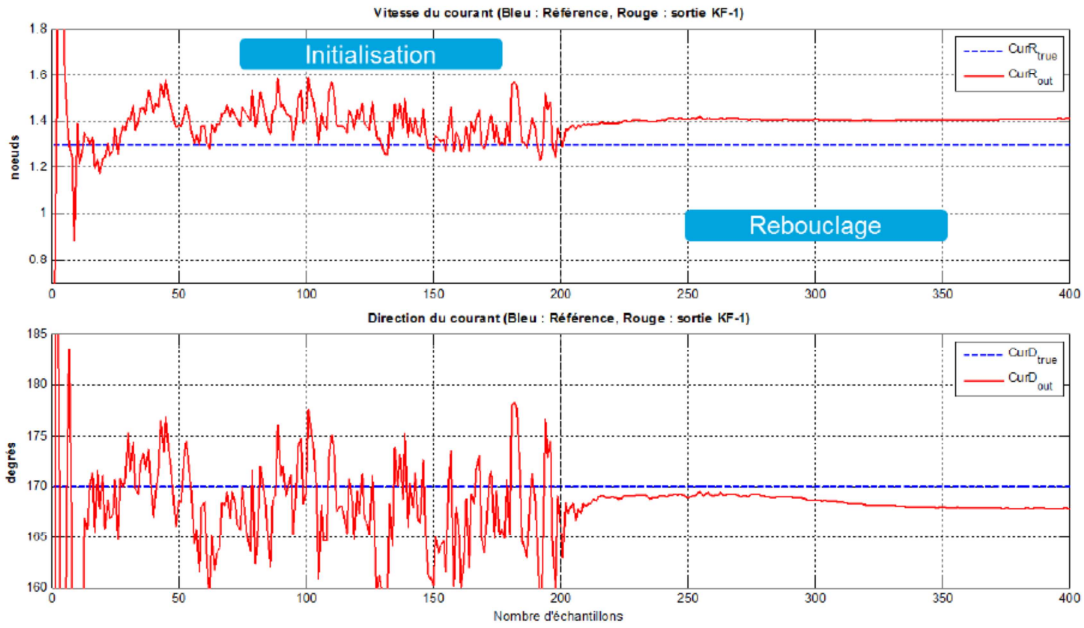


Figure 48: Estimation du courant (cas N°2)

Sur la figure 49, nous nous apercevons que cette mauvaise approximation du courant a un impact direct sur l'estimation du leeway. En effet, durant la phase de rebouclage l'erreur moyenne d'estimation du leeway est de 58 % (cf. tableau 11). Cette erreur est non acceptable et ne permet en aucun cas d'utiliser cette estimation pour l'analyse des performances. Les résultats présents montrent la difficulté d'estimer correctement l'angle de leeway lorsque la formule empirique possède un biais non défini.

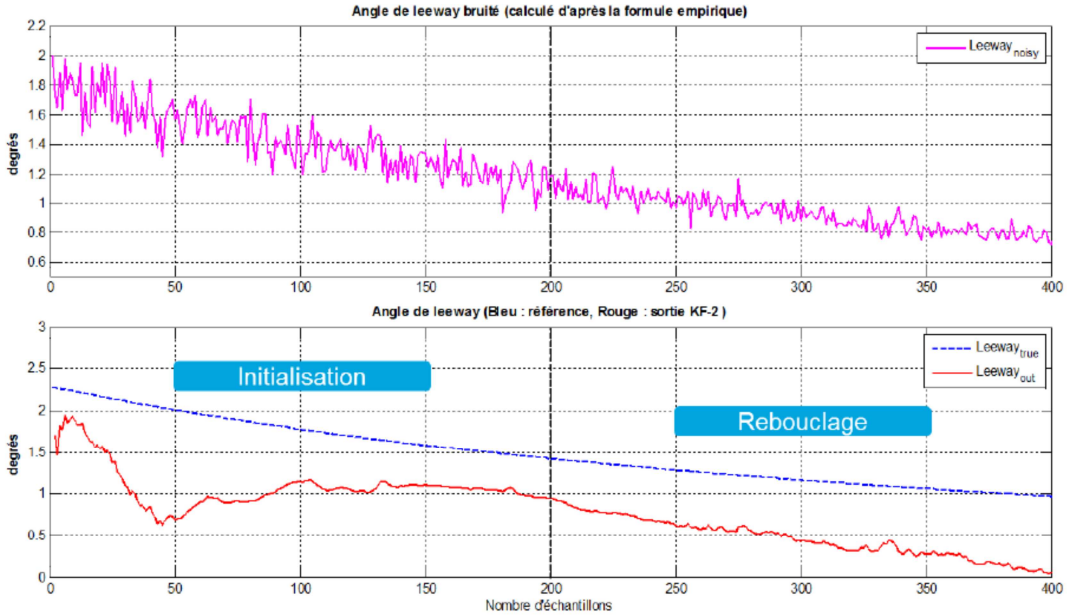


Figure 49: Estimation du leeway (cas N°2)

Paramètres	Unité	Erreur Moy.	Erreur Moy. (%)
CurR	Noeuds	0.106	8.15 %
CurD	Degrés	1.550	0.91 %
Leeway	Degrés	0.684	58.18 %

Tableau 11: Erreur moyenne (cas N°2)

Les résultats obtenus dans le cas d'étude N°1 démontre que notre modèle est capable de converger vers les valeurs du courant avec une précision inférieure à 2% et vers les valeurs du leeway avec une précision de 7%. Cependant dans le cas N°1, nous considérons que la formule empirique du leeway (utilisée pour initialiser le système) est bruitée par les mesures de la vitesse du bateau et de l'angle de gîte mais que la constante k est correct. Or dans le cas d'étude n°2, nous nous apercevons que lorsqu'un biais est introduit sur cette constante, les estimations ne convergent plus vers leurs valeurs de référence. Afin que le modèle converge correctement, il est donc nécessaire que l'erreur sur la formule empirique du leeway reste limitée. En réalité, cette erreur n'est actuellement pas quantifiable puisque nous ne disposons d'aucune référence absolue du leeway sur ce type de bateau. Par conséquent, afin d'améliorer l'estimation du leeway, nous proposons d'utiliser un capteur additionnel qui mesure la vitesse du bateau sur deux dimensions.

L'apport de ce capteur n'est pas présenté dans ce document mais est développé dans l'article de référence fourni en Annexe D ou dans [10]. L'ensemble de la méthodologie pour la mesure du vent présentée dans ce chapitre a été implanté sur une architecture matériel (appelée Sailing Box) et testée sur différents supports du Groupama Sailing Team. Aujourd'hui encore cette Sailing Box est utilisé par Groupama sur ses nouveaux supports à savoir, le DIAM 24 (trimaran utilisé pour le Tour de France à la Voile), Le Classe C (catamaran utilisé pour la petite coupe de l'America) ainsi que sur l'AC-45 (catamaran utilisé pour les World Series anti chambre de la coupe de l'America).

4.4 Conclusion et réflexion

Nous avons présenté dans ce chapitre une des contributions réalisées sur les systèmes pervasifs relative à la mesure du vent réel pour des voiliers de compétition. Ces recherches issues d'une thèse Cifre ont été mises en œuvre sur un système réel embarqué sur différents supports. Ce système ouvert permet de réaliser la même chaîne de traitement qu'une centrale de navigation commerciale en y ajoutant la mesure du leeway (paramètre important pour des monocoques) et surtout en limitant l'empreinte énergétique puisque qu'elle consomme 6 fois moins que la centrale commerciale WTP3 (B&G).

De plus, ces travaux nous ont permis d'acquérir un savoir-faire et une expertise dans le domaine des systèmes embarqués pour le nautisme. Ce savoir-faire nous permet aujourd'hui d'être reconnus et sollicités pour plusieurs projets autour de cet environnement. Une seconde thèse a été soutenue en début d'année autour de la réalité augmentée pour l'aide à la navigation ; cette thèse fait l'objet d'un financement de maturation par la SATT Ouest Valorisation. Nous avons actuellement une thèse Cifre avec la société NKE Marine (dont je suis co-encadrant) qui vise à améliorer encore la mesure du vent dans l'optique d'utiliser cette mesure en entrée d'un nouveau pilote automatique. Celui-ci permettra d'obtenir une qualité de service du pilote plus grande et donc d'améliorer le confort pour le plaisancier lambda et les performances pour le régatier solitaire. Nous sommes également un des

partenaires du projet Voilier du Futur (projet financé par les investissements d'avenir) dans lequel nous intervenons sur le développement d'un gestionnaire d'énergie autonome et temps réel. Bien que ce projet soit aujourd'hui à l'arrêt suite à des problèmes de complément de financement, nous avons bien avancé sur le sujet dans le cadre de la thèse de Mathilde Tréhin.

Nous sommes actuellement en discussion pour le dépôt de 2 projets FUI toujours dans le cadre de cet axe système pervasif et de l'environnement maritime. A contrario des travaux passés et en cours, le verrou principal n'est pas la contrainte de consommation mais plutôt la communication opportuniste. En effet l'idée dans ces 2 projets est de développer des réseaux de capteurs autonomes opportunistes capables de s'adapter aux changements de leur environnement.

Je pense qu'à l'avenir, les travaux sur cet axe vont se développer autour de systèmes enfouis multi contraintes adaptatifs et vont s'orienter vers des problématiques proches de celles rencontrées dans les objets connectés (ou internet des objets). Nous ne devrons plus seulement prendre en compte les aspects fonctionnel et énergétique du système mais également les aspects communications de et vers l'extérieur avec des problématiques de sécurité afin de ne pas compromettre la qualité de service du système global.

Références

[1] Rongere, F. et Kobus, J.-M. (2010). Mesure des trois composantes du vent reel sur support flottant avec correction des mouvements de plateforme. Rapport Technique, Laboratoire de Mecanique des Fluides de l'Ecole Centrale de Nantes.

[2] Lafforgue, D. (2008). Les voiles : de l'experimental au numerique. Mémoire de D.E.A., University of Southampton.

[3] Pedrick, D. et McCurdy, R. (1981). Yacht performance analysis with computers. In Shesapeake Sailing Yacht Symposium, Annapolis, Maryland.

[4] Gentry, A. (1981). Sailboat performance testing techniques. In Proceedings of the Eleventh AIAA Symposium of the Aero/Hydronautics of Sailing, Seattle, Washington

[5] Gentry, A. (2006). The origins of lift. Rapport technique.

[6] Sergent, G. (2011). Upwash study. Rapport technique, Cape Horn Engineering.

[7] Johnson, M., Hodgson, C. et Gauthier, D. (2011). Method and device for determining wind conditions around a sailboat.

[8] Brettle, M. (2001a). Wind direction profiles and yacht sails. Wheather, Volume 56, numéro 5, pages 161-171.

[9] Bijker, J. et Steyn, W. (2008). Kalman filter configurations for a low-cost loosely integrated inertial navigation system on an airship. Control Engineering Practice, Volume 16, numero 12, pages 1509-1518.

[10] Douguet, R. (2014). Optimisation de la mesure et de l'interpretation des performances dans le cadre de la course au large.

Chapitre 5

Conclusion & Perspectives

5.1 Conclusion

Ce document a présenté une partie de mes activités de recherche réalisée durant mes 10 années de Maître de Conférences au sein de l'Université de Bretagne Sud.

Dans un premier temps, mes recherches se sont axées sur l'estimation de la consommation d'architectures logiciel dans la continuité de mes travaux de thèse. Rapidement, je me suis aperçu que cette voie de recherche arrivait à un croisement voire à une impasse. En effet, l'idée la plus prometteuse était de remonter les modèles de consommation vers les niveaux de conception système et cela posait deux problèmes majeurs à savoir : comment extraire les paramètres du modèle de consommation à partir d'une description système et comment modéliser des cibles de plus en plus complexes. Nous avons vu dans le chapitre 2 qu'il était possible d'extraire un certain nombre de paramètre d'un modèle de consommation à partir d'une description système mais ceux-ci étaient souvent généraux (processeur utilisé, fréquence d'horloge). Dès lors que nous voulions obtenir des paramètres moins gros grain (défaut de cache, taux de rupture de pipeline etc…), nous avions besoin du code métier de l'application et donc cela revenait à faire une analyse « classique » de la consommation telle que nous la connaissions déjà (méthodologie tirée de mes travaux de thèse). De plus, le temps d'obtention d'un modèle de consommation d'un processeur par la méthode FLPA (Functional Level Power Analysis) devient, comme pour les méthodes classiques, très important dû à la complexité grandissante des architectures. Aujourd'hui les processeurs, même embarqués, ne se « contentent » plus d'être scalaire ou statiquement parallèle (VLIW) mais peuvent être multi cœurs, dynamiquement parallèles (superscalaire), avoir plusieurs niveaux de cache (L1, L2 & L3) et intégrer des accélérateurs matériels. Tout ceci, rajouté au fait que les systèmes actuels embarquent généralement des systèmes d'exploitation, implique qu'il devient quasiment impossible de réaliser un modèle de consommation à grain fin de telles architectures. Une solution potentielle à ce problème a été exposée dans les perspectives du chapitre 2.

Dans le chapitre 3, j'ai présenté les travaux effectués autour de l'estimation et de l'optimisation de la consommation des interconnexions dans les systèmes sur puce (SoC). Ces travaux ont été les premiers que j'ai co-encadrés à la suite de mon recrutement au sein de l'UBS. Ces travaux ont porté sur deux problématiques à savoir : comment modéliser la consommation d'un réseau d'interconnexions sur une puce en prenant en compte le phénomène de diaphonie capacitive et comment optimiser cette consommation afin de diminuer l'impact de ce réseau sur la consommation globale. Nous avons montré qu'il n'était pas trivial de modéliser en consommation ce réseau d'interconnexions et que de nombreux paramètres impactaient le modèle. Nous avons également démontré que le classement des transitions d'un point de vue consommation était très différent de celui d'un point de vue temporel. Or, jusqu'à notre étude toutes les optimisations de consommation proposées dans la littérature se basaient sur ce classement des transitions ce qui impliquait une sous optimalité de ces approches. Nous avons donc, dans un second temps, proposé de nouvelles optimisations en consommation basées sur notre classement des transitions en consommation. Une de ces méthodes a d'ailleurs fait l'objet d'un dépôt de brevet auprès du CNRS.

Dans le chapitre 4, j'ai présenté le dernier axe de recherche dans lequel je réalise des travaux de recherche à savoir la conception de systèmes pervasifs pour le domaine maritime. Cet axe est aujourd'hui celui où j'ai le plus grand nombre de collaborations et d'activités de recherche. Je n'ai présenté dans ce chapitre que les travaux qui ont fait émerger cet axe de recherche à savoir la

mesure du vent réel pour des voiliers de course au large ; travaux réalisés dans le cadre d'une thèse Cifre avec le Groupama Sailing Team de Franck Cammas. Dans un premier temps, nous avons montré que la mesure de vent, bien que pouvant sembler aisée, est bien plus complexe en réalité. Pour cela, nous avons présenté les différentes perturbations de mesure que nous devons corriger afin d'obtenir une vitesse et un angle pour le vent réel. Nous avons proposé une nouvelle méthode permettant de calculer deux paramètres jusqu'ici mal pris en compte à savoir : le courant et le leeway. Pour se faire notre méthode se base sur de la fusion de données multi capteurs et de filtres de Kalman. Une fois la méthode validée sur des jeux de données simulés, nous avons proposé une implantation matériel d'une chaîne complète de correction du vent, incorporant nos travaux, afin de tester le système dans des conditions réelles à savoir sur un voilier du Groupama Sailing Team. Cette implantation (appelée Sailing Box) ayant été réalisée en tenant compte de l'emprunte énergétique du système, elle aujourd'hui toujours utilisée par l'équipe même sur des supports de voile légère (Diam 24, AC-45…).

5.2 Perspectives

Dans cette dernière section, je vais discuter des perspectives de recherche envisageables sur les thématiques présentées dans ce document. Je ne ferai pas de perspectives à long terme car comme bien souvent ces projections s'avèrent très loin de la réalité. Il y a dix ans, lors de mon recrutement, je n'aurai jamais imaginé travailler sur les systèmes pervasifs et encore moins pour le domaine maritime. La recherche se fait très souvent au gré des opportunités, des gens que nous rencontrons et des envies que nous avons ; il est alors difficile de tracer un sillon en ligne droite. Le fait d'avoir collaboré avec différents collègues du laboratoire est quelque chose de très enrichissant en terme d'ouverture d'esprit et j'aimerais continuer à réaliser un maximum de co encadrement de thèse même après l'obtention de mon habilitation à diriger les recherches.

Sur les travaux présentés dans le chapitre 2, je pense que les perspectives de recherche, pour ma part, sont peu nombreuses du fait de la complexité de développement de modèle de consommation sur les architectures actuelles. Une possibilité pourrait être que l'architecture auto construise son modèle en fonction de son utilisation ; ceci impliquerait le fait de pouvoir intégrer des sondes au sein de l'architecture et de dédié une partie du silicium à l'intelligence de modélisation. Aujourd'hui la solution la plus cohérente serait de mesurer la consommation de l'architecture en fonction de l'application qu'elle exécute. A partir de toutes ces mesures, nous pourrions obtenir une cartographie de consommation en fonction de la classe applicative. Lors d'une estimation, il suffirait de savoir à quelle classe d'application appartient notre programme pour obtenir une estimation de la consommation de celle-ci ; cette estimation même gros grain peut suffire dans la plupart des cas.

Sur les travaux présentés dans le chapitre 3, nous sommes aujourd'hui en train de travailler à l'extension de ceux-ci dans le cadre des réseaux sur puce (NoC). Ces travaux font l'objet de la thèse d'Erwan Moréac qui devrait soutenir en 2017. Nous avons également un projet ANR en cours d'évaluation autour de problématiques proches de celle-ci. En effet, l'enjeu de ce projet ANR est de réaliser le réseau sur puce non plus en purement câblé mais d'en réaliser au moins une partie en utilisant des liaisons sans fil de type RF. L'avantage de la RF réside entre autre dans le fait de pouvoir « broadcaster » une information et donc améliorer, par exemple, le temps nécessaire à la cohérence des caches de l'architecture ou de relancer au plus vite des taches bloquées par une barrière de synchronisation.

Enfin sur les travaux présentés dans le dernier chapitre, j'espère pouvoir relancer les travaux autour de la gestion d'énergie temps réel pour les bateaux. Ces travaux offrent de nombreuses perspectives en termes de transfert industriel surtout au niveau du tissu socio-économique de la Bretagne. J'ai également toujours une thèse Cifre en cours sur les aspects mesure de vent pour l'amélioration d'un pilote automatique avec la société NKE ; ces travaux devraient se terminer vers le début de 2017 et viennent de faire l'objet d'une acceptation de présentation de papier à la conférence High Performance Sailing Yatchs en octobre prochain. Les travaux autour de la réalité augmentée, qui font aujourd'hui l'objet d'une maturation de la SATT Ouest Valorisation, viennent également d'être retenus pour un financement via la fondation BPCE (Banque Populaire) et plusieurs discussions avec des entreprises sont amorcées en vues de l'utilisation de notre technologie. Nous avons également 2 projets en cours de montage toujours dans le domaine maritime mais orientés plus vers les réseaux de capteurs intelligent sans fil et autonome. Ces travaux vont nous permettre de commencer à étudier les recherches menées sur les objets communicants ou Internet des Objets (IOT) car les problématiques visées par ces projets sont très proches de ceux rencontrés dans les objets communicants. De plus, nous avons également obtenu des financements via le CPER afin de développer une constellation de bouées intelligentes dans le but de réaliser une cartographie de vent 3D d'un plan d'eau. Tout ceci est donc en cohérence et permettra de fédérer une équipe autour de ces différentes activités.

ANNEXE A

Energy and power consumption estimation for embedded applications and operating systems

Saadia Dhouib, Eric Senn, Dominique Blouin, Jean-Philippe Diguet and Johann Laurent

Abstract *— This paper presents a methodology that permits to estimate the power and energy consumption of embedded applicationss. Estimation is performed from high-level specifications of the complete system. Power models are built from physical measurements on the hardware platform. Operating system's services are modeled: scheduler / timer interrupt, inter-process communications, devices accesses models are presented. The operating system's energy overhead is expressed as the sum of multiple contributions related to services activated during a run. Our methodology is applied to the modeling of a Xillinx Virtex-II Pro XUP platform, and a Linux 2.6 operating system. The comparison of consumption estimations and measurements for different versions of a multi-threaded MJPEG application shows an error ranging from 1% to 11%. Our methodology and power models have been integrated in a CAD tool, named CAT (Consumption Analysis Toolbox), deployed in the Eclipse IDE and also included in the Open Source AADL Tool Environment, bringing energy estimation capabilities in the AADL design flow.*

Keywords — Power, Energy, Estimation, Model, Operating System, Service, Real-time, Embedded System, Consumption Analysis Toolbox CAT, AADL.

Introduction

Embedded systems are becoming more and more complex. Due to technological improvements, it is now possible to integrate a lot of components in a unique circuit. Nowadays, homogeneous or heterogeneous multiprocessor architectures within SoC (System on Chip) or SiP (System in a Package) offer increasing computing capacities. Meanwhile, applications are growing in complexity. Thus, embedded systems commonly have to perform different multiple tasks, from control oriented (innovative user interfaces, adaptation to the environment, compliance to new formats, quality of service management) to data intensive (multimedia, audio and video coding/decoding, software radio, 3D image processing, communication streaming), and to sustain high throughputs and bandwidths.

One side effect of this global evolution is a drastic increase of the circuits' power consumption. Leakage power increases exponentially as the process evolves to finer technologies. Dynamic power is proportional to the operating frequency. With higher chip densities, thermal dissipation may involve costly cooling devices, and battery life is definitely shortened.

The role of an Operating System (OS) is essential in such a context. Real Time Operating Systems (RTOS) offer a wide variety of services to ease the exploitation of embedded platforms: cooperative and pre-emptive multi-tasking, process management, fixed or dynamic priority scheduling, multi-threading, support for periodic and aperiodic tasks, semaphores, inter-process communications, shared memory, memory, file, and device management. It offers an abstraction of the hardware that permits the reduction of time to design, development, and testing of new systems. It also offers power management services which may exploit low-level mechanisms (low-operating / low-standby power modes, voltage/frequency scaling, clock gating …) to reduce the system's energy consumption [44].

But the Operating System itself has a non negligible impact on the energy consumption. The Operating System's energy overhead depends on the complexity of the applications and the number of services called. In [1] and [2], it was observed that, depending on the application, the energy consumption of an embedded system could rise from 6% to 50%. This ratio gets higher if the frequency and supply voltage of the processor increase. In [3], it is shown that the OS can consume from 1% to 99% of the processor energy depending on the services called. Power consumption is now a major constraint in many designs. Being able to estimate this consumption for the whole system and for all its components is now compulsory. Estimating energy consumption due to the Operating Systems is thus unavoidable. It is the first step towards the application of off-line or on-line power optimization techniques.

It is well-known that high-level optimizations have the greatest impact on the system's final performance. However, they are only possible if estimations can be performed at the earliest levels in the systems' design flow. One of our objectives in the European ITEA project SPICES (Support for Predictable Integration of mission Critical Embedded Systems) [4] is to enrich the AADL model based design flow to permit precise energy and power consumption estimations at different levels in the refinement process. AADL (Architecture Analysis & Design Language) is an input modeling language for the design of real-time embedded systems [5], now commonly used in the avionic and automotive domains. It helps to verify functional and non-functional properties of the system, from early analysis of the specification to code generation for the hardware platform [6, 7, 8].

A lot of work has been done on power consumption estimation at different abstraction levels in embedded systems design. Many approaches deal with low level models, and are dedicated to the analysis of hardware components, or part of hardware components. They are not usable for complex systems. In this paper, we will focus on approaches intended to assess

the power and energy consumption of a complete system, including its operating system. In fact, the major contribution of this paper, compared to our former publications, is the proposition of a methodology for estimating the consumption of a complete embedded system, and that includes a multi-threaded application, a real-time operating systems with different services, and a heterogeneous hardware platform. Different consumption models associated to the considered operating system services are presented as well.

In the frame of the SPICES project, our consumption analysis tool and power models are coupled with timing analysis tools working at different abstraction levels and with various precisions. All those tools can be found in the OSATE (Open Source AADL Tool Environment) [27] and TOPCASED [35] tools suite. Energy consumption estimations accuracy thus depends on the error introduced by those timing analysis tools. In our paper, we have chosen to present our results considering measured execution time when needed, in order to isolate the error introduced by our power models, from the error due to the timing analysis tools. Hence, the performances of our approach should appear more clearly to the reader. Static or dynamic timing analysis may be performed with tools related to the SPICES project (see [4]: SoftExplorer [30], BIP [36], TINA [37], PathCrawler [38], ADES [39], MAST [40]) and AADL (Cheddar [41]), SystemC simulations at different levels, including the operating system (Scope [42], AADS [43]).

The remainder of this paper is organized as follows. We present related work in section 2. Section 3 describes our multi-layer estimation methodology. In section 4, we present power models on which estimation is based. Section 5 presents our consumption analysis toolbox which integrates the methodology and related power models. In section 6, we evaluate our methodology by estimating the energy consumption of three multithreaded versions of a MJPEG encoder. We conclude the paper in section 7.

State of the art

In [9], energy estimation of embedded Operating Systems relies on micro-architectural cycle-accurate simulation. The software energy estimation is performed at four granularity levels: atomic function, routine, service, and kernel execution path. A full system instruction level simulator (based on *Skyeye* [10]) provides an instruction and address trace that is an input of the micro-architectural simulator, which includes power models of micro-architectural components. The *Strong ARM* architecture platform running Linux 2.4 is considered. This approach inherits the drawbacks of micro-architectural level power analysis, as performed in the tools *Wattch* [11] or *SimplePower* [12]. Firstly, cycle-level simulations may prove very time consuming for large programs. Secondly, they necessitate a low-level description of the architecture which is often difficult to obtain for off-the-shelf processors. Hardware devices (Ethernet, external memories, Compact Flash /Disks, External controllers …) are not considered either, even if they can represent the major part of power consumption of embedded systems.

In [13], Fournel et al. presented *eSimu*, a performance and energy consumption simulator for deeply embedded hardware platforms, again built upon *Skyeye*. This approach is based on cycle accurate simulations of complete hardware platforms executing the real application code. Quantitative energy data are gathered at the battery output and are translated into per instruction and peripheral event energy figures. An *ARM9* based embedded system is considered. The Operating System overhead is, however, not identified. Energy consumption is reported at the source code level regardless of any OS or driver implementation.

In [14] energy estimation is performed at the system level. FPGA accelerated simulation technologies are used to speed-up the system simulation. Energy estimation currently relies on spreadsheet to compute power consumption from the accelerated simulation results, but

analytical models might be used as well. One major drawback with this approach is the difficulty to validate and calibrate the simulator against actual implementations on silicon, which grows with the complexity of processors core. Tedious porting work is necessary to support new target model, since a large number of power measurements are necessary for every component in the processor core.

Byanes et al. [15] proposed an execution driven simulation testbed that measures the execution behavior and power consumption of embedded applications and RTOS, by executing them on an accurate architectural model of a microcontroller with simulated real-time stimuli. The power consumption estimation relies on instruction based techniques (10-15% error). The so-called *Instruction-Level Power Analysis* (ILPA) [16] relies on current measurements for each instruction and couple of successive instructions. Even if it proved accurate for simple processors, the number of measures needed to obtain a model for a complex architecture would become unrealistic [17]. In [15] the inter-instruction overhead is not precisely modelled. Three different RTOS have been compared: *µCOS-II, Echnida*, and *NOS* [2]. However, the overhead of the OS is considered globally and not on a per service approach. The difference between the application and OS consumption is not considered.

In [18], simpler models of OS power consumption, with few parameters, are proposed. Li and al. considered OS service routines as the fundamental unit of OS execution and measured that they have similar and predictable power dissipation behaviors across various benchmarks. They found strong correlation between the power and the *Instruction Per Cycle* (IPC) metric, and developed a model that exploits this correlation. *Instruction Per Cycle* is similar to the *parallelism rate* that we introduced in our own approach [19, 20] for modelling the power consumption of complex processors. It is related to the fact that in modern high performance superscalar processors, a dominant part of the power is consumed by circuits used to exploit Instruction Level Parallelism.

Vahdat et al. [21] conducted a general study on aspects of Operating System design and implementation to improve energy efficiency. They investigated low power modes of embedded devices and proposed energy efficient techniques to use operating system functionalities.

Acquaviva et al. [1] proposes a methodology to analyse the OS energy overhead. They measured the energy consumption of the *eCos* Real Time Operating System running on a prototype wearable computer, HP's *SmartBadgeIII*. They analysed the energy impact of the RTOS both at the kernel and the I/O driver level, and the influence of different factors like I/O data burstiness and thread switch frequency. They particularly focused on the relationship between the energy consumption and processor frequency. The authors actually analyzed, but did not model, the energy consumption for the internal services and I/O drivers of the operating system.

Dick et al. [3] analyzed the energy consumption of the μC/OS RTOS when running several embedded applications. They targeted a *Fujitsi SPARClite* processor based embedded system. This work presents only an analysis of RTOS policies on embedded system power consumption. The authors did not develop an energy consumption model.

In [22], a methodology is proposed to characterize embedded OS energy systematically. The energy consumption of OS services and primitives is modelled. A first analysis provides energy characteristics, which is a set of essential components of the operating system used to characterize its energy consumption. Then, macro-modelling gives quantitative macro-models for the energy characteristics. The OS considered are μCOS and Linux. Two low-level energy simulation tools (Sparcsim, EMSIM) were used for the characterization instead of direct current measurement. This approach is thus limited by the accuracy of those energy simulators. Only internal OS mechanisms are considered, the energy consumption of external

I/O drivers was not investigated.

To allow for a fast and fruitful exploration of the design space at high-levels in model driven approaches, power consumption estimations have to be completed in a reasonable delay. We have introduced the Functional Level Power Analysis (FLPA) methodology which we have applied to the building of high-level power models for different hardware components, from simple RISC processors to complex superscalar VLIW DSP [18, 19], and for different FPGA circuits [20]. Our methodology allows for a fast and precise estimation of the power consumption of complex systems: it does not rely on a simulation, involving more or less low-level models, but on a static analysis of the specification, and possibly a profiling of the application. In fact, simulation may be impossible if the actual code of the application is not known in the early stages of the design. Our methodology allows to estimate the power consumption of heterogeneous architectures, since it may be applied to the building of power-models for every component in an embedded system. Today we extend this approach by considering heterogeneous architectures and the overhead due to the operating system services and the use of peripherals in addition to the own consumption of applications; the aim is to offer a reasonable trade-off between estimation speed and accuracy.

Multi-layer power and energy estimation methodology

Our approach consists in separately estimating the different sources of power consumption. However, the different components of embedded systems usually share common power supplies that prevent complete differentiation. For instance, in the *Xilinx Virtex II* board (*XUP*), the Compact Flash memory, the SDRAM and the FPGA I/O are powered by the same 2.5V power supply. Thus, we have adopted an incremental approach where we consider that energy can be summed over the hyper period of the real-time system, contrary to power, which is usually localized and non-summable over different physical components.

Figure 1 presents how power and energy consumption are estimated in the case of the 2.5V power supply of the XUP board.

Figure 1.a represents real power and energy consumption while Figure 1.b illustrates our approach and how we distinguish and estimate the different sources of energy consumption. Of course, the objective is to obtain an estimated area as close as possible to the real area.

We first consider what we call P_{ground}, the power consumption of all the components when the system, without OS, is not executing any application. This power consumption can be quite important especially in the case of embedded systems implemented on FPGA. Then we add the energy contribution of each task by considering the overhead related to peripherals and OS services.

Figure 2 shows how the energy consumption of one task is estimated. It is the sum of multiple contributions represented by different areas, called layers. One layer Li corresponds to the consumption of one part of the system for task i. The energy consumption of the whole system is the addition of the energy consumption of every task. The average power is computed by considering the estimated energy and the application specific period.

Our approach is based on high-level power modelling detailed in the next section, it basically consists in applying estimation, for every task in the system, with the following order:

L1. E_{ground}, it corresponds to the task's basic energy consumption and depends on the basic power consumption P_{ground} and the execution time of the task.

$$E_{ground} = P_{ground} \times T(\tau) \quad (1)$$

P_{ground} is based on a simple model built from physical measurements of the targeted platform power consumption in the corresponding "idle" configuration. The task's

execution time information is estimated by direct measurements or by timing estimation tools.

L2. δE_{τ}, task intrinsic contributions. It corresponds to the energy consumption overhead of the task, considered standalone.

$$\delta E_{\tau} = \delta P_{\tau} * T(\tau) \tag{2}$$

The estimation of δP_{τ} relies on high-level power models of the targeted processor, and includes Cache and RAM accesses.

L3. $\delta E_{timer\_interrupt}$, this is the basic OS energy consumption incurred by timer ticks tied to the scheduler.

$$\delta E_{timer\_interrupt} = \delta P_{timer\_interrupt} \times T(\tau) \tag{3}$$

$\delta P_{timer\_interrupt}$ is the same for all the tasks and for a chosen system configuration (processor, bus and tick timer frequencies). The timer interrupt energy overhead ($\delta E_{timer\_interrupt}$) is variable since it depends on the execution time T of the task (τ).

L4. $\delta E_{scheduler}$, scheduler overhead. It includes context switches and scheduling operations, which are estimated individually for each task.

$$\delta E_{scheduler} = \delta P_{scheduler} \times \delta T_{scheduler} \tag{4}$$

L5. δE_{IPC}, energy due to communication and synchronization services.

$$\delta E_{IPC} = \delta P_{IPC} * \delta T_{IPC} \tag{5}$$

L6. δE_{device_access}, energy overhead related to peripheral device accesses (Flash, Ethernet or other specific controllers).

$$\delta E_{device} = \delta P_{device} * \delta T_{device} \tag{6}$$

Power Models

As previously stated, our power models are based on the *Functional Level Power Analysis* approach. FLPA implies the decomposition of a complex system into functional blocks that are independent with respect to power consumption [23]. Parameters are identified that characterize the way the functional blocks will be excited by the input specification. A set of measurements is performed, where the system consumption is measured for different values of the input parameters. Consumption charts are plotted and mathematical equations are computed by regression. This method is interesting because it links low-level measures and observations of the physical implementation, with high-level parameters from earlier steps in the design flow.

The model output is compared with measured values over a large set of realistic applications. This allows for defining the maximal and average errors introduced by the model. In the following section, we first present power models of standalone tasks running on embedded processors. We then introduce operating system services models such as timer interrupt, IPC, device accesses, which are linked to embedded devices such as the memory and device controllers.

Power models of standalone tasks

Standalone tasks consume energy when running on processors and when accessing the memory in case of cache misses. Power models of embedded processors and memories have been presented in previous works [33][34]. Different power models have been developed so far, for different architectures, from the simple RISC (ARM7, ARM9) to much more complex architectures (the super scalar VLIW DSP TI-C62, C64, and C67), and also for low-power processors (the TI-C55 and the Xscale). Important phenomena are taken into account, like cache misses, pipeline stalls, and internal / external memory accesses. The average error, observed between estimations and physical consumption measurements, for a set of various

algorithms (FIR filter, LMS filter, Discrete Wavelet Transform (DWT) with different image sizes, Fast Fourier Transform (FFT) 64 to 1024 points, Enhanced Full Rate (EFR) Vocoder for GSM, MPEG1 decoder, MJPEG chain …) is lower than 5% at the assembly level, and lower than 10% from the C-code. In the case of the PowerPC 405 [24], the model comes with an accuracy which is better than 5% (the average maximum error found between the physical measurements and the results of a law). The average error is 2%. Input parameters are: the processor frequency and the frequency of the bus it is connected to, the configuration of the memory hierarchy associated to the processor's core (i.e. which caches are used (data and/or instruction) and where the primary memory is (internal/external to the FPGA)), and the cache miss rate of the task.

Power models of embedded OS services

We have developed power and energy models of timer interrupt, IPC, Ethernet and Compact Flash, for the Linux 2.6 operating system ported to the Xilinx Virtex II pro development board (Section 6.1). Current variations on the 2.5 V power supply connected to the FPGA I/O, SDRAM and Flash disk; and on to the 3.3V power supply connected to the Ethernet physical controller, are measured. The Agilent DC Power Analyser [25] is used to source and measure DC voltage and current into the XUP board. The Power Analyzer is a mainframe that has four slots to accept one to four DC Power Modules. Each DC Power Module has a fully integrated voltmeter and ammeter to measure the actual voltage and current being sourced out of the DC output into the XUP board. The digitizer in each module runs at 50 kHz and captures 4096 samples per trace; the upper limit on samples per trace depends on the memory configuration of the power analyzer.

Power model of OS timer interrupts

Every timer tick, the scheduler_tick() function is called to evaluate the runnable processes. To estimate the energy overhead incurred by timer interrupts, we have executed several

computing intensive programs with and without OS. We have observed the same power overhead for all programs. We show the results for two programs, the first one is a discrete cosine transform (DCT) applied to 8*8 pixels blocks stored in RAM. The second is a quantizer that rounds off the DCT coefficients according to a quantization matrix. Each function is repeatedly executed on the same frame, so that the cache miss effect is attenuated. We have executed the programs with different processor and OS tick timer frequencies. With Linux 2.6, we can tune the tick timer frequency to 100, 250, 300 or 1000 Hz. We measured the power consumption of each program running with and without OS, and for each system configuration. Energy was computed using timing information taken by direct measurements.

Figure 3 shows the energy overhead of the timer interrupt. Energy overhead is computed as follows:

$$\delta E_{timer\_interrupt} = E(\tau_{withOS}) - E(\tau_{w/oOS}) \tag{7}$$

Where τ_{withOS} corresponds to the program execution with OS and $\tau_{w/oOS}$ corresponds to the program execution without OS.

The power consumption overhead is modelled as follows:

$$\delta P_{timer\_interrupt} = \frac{E(\tau_{withOS}) - E(\tau_{w/oOS})}{T(\tau_{withOS}) - T(\tau_{w/oOS})} \tag{8}$$

Where T is the measured execution time of the corresponding program. We observed that power and energy overhead variations are the same for every program, and depend on the tick timer and CPU frequencies. The power overhead model is written as follows:

$$\begin{pmatrix} \delta P_1 \\ . \\ . \\ \delta P_n \end{pmatrix} = \begin{pmatrix} \alpha_1 & \beta_1 \\ . & . \\ . & . \\ \alpha_n & \beta_n \end{pmatrix} . \begin{pmatrix} X \\ 1 \end{pmatrix} \tag{9}$$

Where P_i corresponds to power consumption related to the CPU frequency i value; X is the

timer tick frequency (Hz); α_i is a coefficient associated to the tick timer frequency.

Power model of IPC

Inter-process communications allow threads in one process to share information with threads in other processes, and even processes that exist on different hardware platforms. In IPC, the RTOS explicitly copies information from a sending process's address space into a distinct receiving process's address space. Examples of IPC mechanisms are pipes and message passing through mailboxes or sockets (remote interprocess communication).

To model IPC power consumption, we have executed programs that repeatedly use an IPC mechanism, with different values for parameters such as the amount of data sent and received, the OS tick timer frequency and the processor frequency. For example, for measuring the energy cost of message queue read or receive, we have repeatedly executed programs containing one software process and two threads. The first thread basically sends messages of variable length, using the system function mq_send(), to a message queue. The second thread retrieves the messages using the system function mq_receive(). Communications are performed within the same process to avoid process context switching. We have also executed test programs with a high real time priority to avoid preemptive context switch. Our measurements show that only the processor frequency and the message size influence the IPC consumption. Thus, the power model of IPC mechanisms is:

$$P_{IPC} = aF_{cpu} + b \tag{10}$$

The detailed power model of message queues, shared memory and pipes is detailed in Table 1.

The energy model, given the processor frequency, is a function of the message size:

$$E_{IPC}i = c_i X^{p_i} + d_i \tag{11}$$

Where $E_{IPC}i$ is the energy consumption per byte transmitted for frequency i, c_i, p_i and d_i are coefficients of the model. The energy model of the Mqueue mechanism is depicted in Table 2. We present energy laws, as well as the average model error rates for the Mq_send and Mq_receive routines. The error rate is computed following the equation:

$$\sum_{i=1}^{n} \frac{1}{n} \frac{\left| \hat{E}_i - E_i \right|}{E_i} \tag{12}$$

Where $\hat{E}_i$ is the estimated energy and E_i is the measured value.

In the case of remote communications, we have presented the Ethernet model in [26]. The model has the same parameters as the local communication model, with an additional parameter, which is the protocol type (TCP or UDP). Our models come with an accuracy that is better than 9% (the average maximum error found between the physical measurements and the estimations). The average error is 8%.

Power model of compact flash accesses

We selected a standard storage device, the compact flash, as a representative example implemented in most embedded systems. As a first step, we identified the key parameters that can influence the power and energy consumption of compact flash accesses. Then we conducted physical power measures on the XUP board. Measurements were realized with different testbenches containing RTOS routines accessing the compact flash.

We have modelled two different Linux system calls. The first model concerns buffered I/O which are the default Linux I/O operations. When the I/O is buffered the compact flash does direct memory access from/to the kernel cache, and not from/to the user space source/destination buffer allocated by the user application. The second model concerns self caching I/O. In this case, the application will keep its own I/O cache in user space (often in

shared memory), so it does not need any additional lower level system cache. This is done using the option O-Direct when reading or writing data.

Power and energy models have the same form as that of equations (10),(11). We have noticed that buffered I/O consumes more power, but less energy than self caching I/O. This is due to the fact that when using buffered I/O, more resources are solicited (RAM, CPU caches) that increase the power consumption. On the other hand, buffered I/O is more efficient because it allows programs to reduce dramatically the number of I/O operations during the runtime of the system. Consequently, it consumes less energy.

CONSUMPTION ANALYSIS TOOLBOX

Our energy estimation methodology and power models have been integrated in a Computer Aided Design (CAD) *Consumption Analysis Toolbox* (CAT). The toolbox combines a set of power estimation models with a system architecture model to provide system-level power consumption analysis. CAT has been used on our case study to compute the estimated power consumption. CAT runs on the Windows and Linux platforms and is deployed on the Eclipse Integrated Development Environment (IDE). The central part of CAT is a Domain Specific Language (DSL) that was defined to describe system architectures from a power analysis perspective. It also serves as a communication layer between the CAT application tiers to exchange the modelled system data. CAT can also be used in conjunction with the *Open Source AADL Tool Environment* (OSATE) [27], and the Toolkit in Open source for Critical Aeronautic Systems Design (TOPCASED) [35]. In this version of the toolbox, the AADL instance model editor can be used instead of the CAT model object editor. CAT may be downloaded with related documentation on [45].

Figure 4 presents the component based AADL design flow. The AADL component assembly model contains all the components and connection instances of the application, and

references the implementation models of the components instances from the AADL models library. The AADL target platform model describes the hardware of the physical target platform. This platform is composed of at least one processor, one memory, and one bus entity to home processes and threads execution. The AADL deployment plan model describes the AADL-PSM (Platform Specific Model) composition process. It defines all the binding properties that are necessary to deploy the processes and services models of the component-based application on the target platform. All those models are combined to obtain the AADL-PSM model of the complete component-based system. The final implementation of the system is obtained afterward through model transformations and code generation. Deployment of the application on the hardware platform consists in binding processes to memories, threads to processors, and connections to busses. The Operating system process and threads are also bound to the memory and processor. The resulting Platform Specific Model is then analysed and parameters influencing power and energy consumption are automatically extracted by CAT. The power and energy consumption is computed afterward. More details on the AADL design flow and power analysis may be found in [24] and [28].

In the next section, we show how the CAT tool performs power and energy estimation on different multithreaded versions of an MJPEG system.

CASE STUDY

This section illustrates the application of our methodology on a Motion JPEG (MJPEG) encoder. Starting from an AADL high level description of the MJPEG system, the CAT tool extracts OS-related parameters and estimates energy consumption for the whole system. Three different MJPEG scenarios, with different communication mechanisms, are considered. Estimations are finally compared to the measured consumption to validate our methodology.

Experimental framework

The platform is a XUP Virtex-II pro development board, featuring a 256MB SDRAM memory, a 512 MB Compact Flash, and an Ethernet interface. The Virtex-II pro includes two PowerPC 405 hard cores, with memory management units, data and instruction caches. Many I/O controllers are also integrated in the FPGA, such as audio, Ethernet, UART, compact flash and SDRAM. This device has been selected as a usual low-cost solution for a configurable SoC including hard-coded IPs. The operating system is Linux 2.6 which introduces a new low-latency preemptive scheduler whose execution time is not affected by the number of tasks being scheduled. Along with POSIX threads, it provides POSIX signals, POSIX message queues and POSIX high-resolution timers as part of the mainstream kernel.

MJPEG application scenarios

Motion JPEG is an informal name for multimedia formats where each video frame or interlaced field of a digital video sequence is separately compressed as a JPEG image. It is often used in mobile appliances such as digital cameras. We have specified three different AADL models of the MJPEG application. Each model corresponds to a scenario where tasks are organized and communicate differently. Table 3 summarizes the three MJPEG scenarios.

Scenario 1 (Figure 5) is constituted of one process containing threads synchronized with signals. Scenario 2 contains two processes communicating through mqueues. Each process includes a set of threads synchronized with signals (modelled as connections between thread event ports in Figure 6).

Data transfers between threads are modelled as AADL data port connections. *Image_Acquisition* periodically gets frames from the Compact Flash and transfers them to *RGB2YUV*. Thread *RGB2YUV* divides a frame into a set of 8*8 pixels blocks, and converts RGB blocks to YUV blocks. The remaining compression operations are separately fulfilled on

Y, U and V frames. DCT performs transformations on Y frames, and then sends the resulting data to the MJPEG_part_2 process through a mqueue. Scenario 2 uses 3 mqueues, each one corresponding to Y, U or V frames.

To specify IPC services using AADL, we have extended the language with new packages [29]. Mqueues are implemented in the RTOS kernel memory space; services of mqueues are implemented as server subprograms of the mqueue thread. Data received from a mqueue is further quantized and then compressed with a loss-less algorithm, a variant of Huffman encoding. Finally thread Rebuild_Image concatenates the obtained (Y, U, V) frames with the JPEG header into a compressed image.

Scenario 3 is organised as scenario 2, but uses a remote communication mechanism (socket). The application is distributed on two XUP platforms connected through Ethernet as shown in Figure 8.

Energy estimation on the MJPEG system

We applied our estimation methodology, following the different steps presented in section 3, for scenarios 2 and 3. We start with the AADL specification that provides the mapping of each thread on the target architecture. The CAT tool first computes the power consumption of standalone tasks using the PowerPC 405 model presented in [27]. That corresponds to consumption layers L1 and L2 in our approach: P_{ground} is included in the estimated task consumption given by the model. The estimation is based on the SoftExplorer tool [30], integrated in CAT, which can analyse C programs specified for each thread according to the processor model. Parameters of the model are extracted from the AADL specification of the system to analyze. The tool extracts input parameters for the processor's power model which are data and instruction cache miss rates and processor and bus frequencies. Cache miss rates are obtained using the cache profiler tool Cachegrind, from the Valgrind tool suite [31]. From

the timing information of each task, here obtained by direct measurements, the tool estimates the thread energy consumption. Timing information can also be obtained from timing analysis tools currently being developed in the SPICES project, such as PathCrawler [32] which is specialized to be used for worst-case/best-case execution time prediction.

Secondly, the CAT tool adds the energy overhead due to the timer interrupts and the scheduler. It corresponds to consumption layers L3 and L4 respectively. These estimations are computed with input parameters of the RTOS model. Relevant information for these estimations, which are extracted from the AADL specification, are the processor and the tick timer frequencies.

In a third step, the CAT tool adds the energy overhead due to IPC corresponding to the consumption layer L5. In scenario 2 (resp. 3), the communication mechanism is message queues (resp. Ethernet). Following the mqueue (resp. Ethernet) model, the tool extracts, from the AADL specification, parameters such as message (resp. IP packet) size, amount of data sent or received, processor frequency and protocol type (in case of Ethernet); and it computes the energy overhead.

Finally, CAT adds the energy overhead of device accesses related to the consumption layer L6. Threads *Image_Acquisition* and *Rebuild_Image* have access to the Compact Flash memory (Input parameters of the Compact Flash model are the processor frequency and the amount of data read or written). Table 4 shows the details of the estimation for each task in the system. γ is the cache miss rate of task τ_i.

The estimation approach we propose has some fitting error with respect to the measured energy values. We use the following average error metric:

$$\frac{\left|\hat{E}_i - E_i\right|}{E_i} \tag{13}$$

Where $\hat{E}_i$ is the estimated energy and E_i is the measured value. Table 5 shows the error rate of the estimation approach for the MJPEG multithreaded application scenarios. While more power consumption sources are considered when the complexity increases, the cumulative error increases as well. The higher error for the third scenario (part 1 and 2 respectively corresponds to the first and second platform) comes from the Ethernet communications consumption model, for which the maximum error is 9%.

Conclusion

We have presented a methodology that permits a system level estimation of the power and energy consumption of complete embedded applications. Our methodology comes with a set of power models for the hardware platform on which the application is deployed, and the operating system which operates the platform. The operating system's consumption overhead is view as the sum of multiple contributions corresponding to every activated service, such as scheduler/timer interrupts, inter-process communications, and peripheral devices access. The analysis is incremental: for each task the power and energy consumption is estimated before the operating system's overhead.

Our methodology has been applied to the modeling of the Xillinx XUP Virtex-II Pro platform and Linux 2.6 OS. Estimations were performed on three multi-threaded versions of the MJPEG application, involving different communication mechanisms and the corresponding OS services, and one or two XUP boards. Estimations were compared to physical consumption measurements: the error between our estimations and the measured consumption goes from 1% to 11%.

Our methodology, and the associated power models, has been integrated in the tool CAT

(Consumption Analysis Toolbox), deployed in the Eclipse IDE. CAT comes with a Domain Specific Language (DSL) to describe the system architectures from a power analysis perspective. CAT was also integrated in OSATE, the Open Source AADL Tool Environment, and TOPCASED.

Future works will include the development of power models for operating system's services not considered yet, especially with a strong impact on the memory stress (memory virtualization, paging and swapping mechanisms). Dedicated power management mechanisms have also to be considered. Future works include the use of our methodology on several different multi-threaded real-time applications. An automatic measurement platform will be build, to automatize the measure of power and energy consumption on different embedded systems, that will permit the validation of our approach on a larger set of various applications. Those works will be conducted in the French ANR project Open-PEOPLE [46]. In the frame of this project, we are planning the modelling of other hardware target platforms, and especially the inclusion of FPGA power models, which we have already developed for standalone components, in our estimation framework for a complete heterogeneous system.

REFERENCES

[1] A. Acquaviva, L. Benini, and B. Ricc´o. “Energy characterization of embedded real-time operating systems”, In Proceedings of the Workshop on Compilers and Operating Systems for Low Power (2001).

[2] K. Baynes, C. Colins, E. Fiterman, B. Ganesh, P. Kohout, C. Smit, T. Zhang, and B. Jacob, “The performance and energy consumption of three embedded real-time operating systems”, ACM International Conference on Compilers, Architecture and Synthesis (2001).

[3] R. P. Dick, G. Lakshminarayana, A. Raghunathan, and N. K. Jha, “Power analysis of embedded operating systems”, In Proceedings of the 37th Conference on Design Automation, pp. 312–315,.

[4] The SPICES ITEA Project Website.. http://www.spices-itea.org/

[5] P. Feiler, B. Lewis, and S. Vestal. The sae architecture analysis & design language (AADL) A standard for engineering performance critical systems. In IEEE International Symposium on Computer-Aided Control Systems Design (2006), pp. 1206–1211.

[6] T. Vergnaud, “Modélisation des systèmes temps-réel embarqués pour la génération automatique d applications formellement vérifiées,” Ph.D. dissertation (2006), Ecole Nationale Supérieure des Télécommunications de Paris, France.

[7] A. Rugina, K. Kanoun, and M. Kaniche, “AADL-based dependability modelling,” LAAS, Tech. Rep., (2006).

[8] J. Hugues, B. Zalila, and L. Pautet, “Rapid prototyping of distributed real-time embedded systems using the aadl and ocarina,” in Proceedings of the 18th IEEE International Workshop on Rapid System Prototyping (2007). IEEE Computer Society Press, pp. 106–112.

[9] X. Zhao, Y. Guo, H. Wang, X. Chen, “Fine-Grained Energy Estimation and Optimization of Embedded Operating Systems”, International Conference on Embedded Software and Systems Symposia (2008), pp. 90 – 95.

[10] Y. Chen, “The Analysis and Practice on Open Source Embedded System Software--Based on SkyEye and ARM Developing Platform”,: Beihang University Press (2004).

[11] D. Brooks, V. Tiwari, and M. Martonosi, “Wattch: A framework for architectural-level power analysis and optimizations,” in Proc. International Symposium on Computer Architecture ISCA’00, 2000, pp. 83–94.

[12] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. Irwin, “The design and use of SimplePower: A cycle accurate energy estimation tool,” in Proc. Design Automation Conf (2000), pp. 340–345.

[13] N. Fournel , A. Fraboulet, P. Feautrier, “eSimu: a Fast and Accurate Energy Consumption Simulator for Real Embedded System”, IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (2007), pp.1 – 6

[14] D. Sunwoo, H. Al-Sukhni, J. Holt; D. Chiou. “Early Models for System-Level Power Estimation”, Eighth International Workshop on Microprocessor Test and Verification (2007), pp. 8 – 14

[15] K. Baynes, C. Collins, E. Fiterman, G. Brinda, P. Kohout, C. Smit; T. Zhang, B. Jacob "The performance and energy consumption of embedded real-time operating systems", IEEE Transactions on Computers, Volume 52, Issue 11, (2003) pp. 1454 – 1469

[16] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: a first step towards software power minimization," IEEE Trans. VLSI Systems (1994), pp. 437–445.

[17] B. Klass, D. Thomas, H. Schmit, and D. Nagle, "Modeling inter-instruction energy effects in a digital signal processor," in Proc. of the Power Driven Microarchitecture Workshop in ISCA'98, (1998).

[18] T. Li, L.K. John "Run-time modeling and estimation of operating system power consumption", ACM SIGMETRICS'03 (2003), San Diego, USA.

[19] N. Julien, S. Gailhard, and E. Martin, "Low power synthesis methodology with data format optimization applied on a DWT," Journal of VLSI Signal Processing (2003), pp. 195–211.

[20] E. Senn, N. Julien, J. Laurent, and E. Martin, "Power Consumption Estimation of a C Program for Data-Intensive Applications", the 12th IEEE International Workshop on Power And Timing Modeling, Optimization and Simulation (2002).

[21] A. Vahdat, A. Lebeck, and C. Ellis, "Every joule is precious: A case for revisiting operating system design for energy efficiency", Ninth ACM SIGOPS European Workshop (2000).

[22] T.K. Tan, A. Raghunathan, N.K. Jha, "Energy macromodelling of embedded operating systems", ACM Transactions on Embedded Computing Systems (2005), vol. 4, N° 1, pp 231-254.

[23] J. Laurent, N. Julien, E. Senn, and E. Martin, "Functional Level Power Analysis: An efficient approach for modeling the power consumption of complex processors," in Proc. Design Automation and Test in Europe DATE, (2004).

[24] E. Senn, J. Laurent, E. Juin, and J.-P. Diguet, "Refining power consumption estimations in the component based AADL design flow," in ECSI Forum on specification & Design Languages (2008), pp. 1206–1211.

[25] Agilent, "N6705A DC Power Analyzer Service Guide", http://home.agilent.com (2007)

[26] S. Dhouib, J.-P. Diguet, E. Senn, and J. Laurent, "Energy models of real time operating systems on FPGA," in Euromicro International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (2008).

[27] The Open Source AADL Tool Environment (OSATE). http://aadl.sei.cmu.edu/aadlinfosite/OpenSourceAADLToolEnvironment.html.

[28] E. Senn, J. Laurent, and J-P. Diguet, "Multi-Level power consumption modelling in the AADL design flow for DSP, GPP, and FPGA", International Workshop on Model Based Architecting and Construction of Embedded Systems (2008).

[29] S. Dhouib, E. Senn, J.-P. Diguet, J. Laurent, and D. Blouin, "Model driven high-level power estimation of embedded operating systems communication services," in The 6th International Conference on Embedded Software and Systems (2009).

[30] E. Senn, J. Laurent, N. Julien, and E. Martin, "Softexplorer: Estimation, characterization, and optimization of the power and energy consumption at the algorithmic level," in *PATMOS* (2004).

[31] N. Nevercote and J. Seward, “Valgrind: A framework for heavyweight dynamic binary translation,” in Programming Language Design and Implementation (PLDI’07), 2007, pp. 89–100.

[32] PATHCRAWLER Home. http://www-list.cea. fr/labos/fr/LSL/test/pathcrawler/index.html.

[33] N. Julien, J. Laurent, E. Senn, E. Martin, "Power Consumption Modeling of the TI C6201 and Characterization of its Architectural Complexity", IEEE Micro, Sept/Oct 2003, Special Issue on Power- and Complexity-Aware Design, Guest Editors: Dave Albonesi, Pradip Bose, Diana Marculescu.

[34] E. Senn, J. Laurent, N. Julien and E. Martin, "SoftExplorer: Estimating and Optimizing the Power and Energy Consumption of a C Program for DSP Applications", the EURASIP Journal on Applied Signal Processing, Special Issue on DSP-Enabled Radio, vol 2005, no. 16, 1 September 2005.

[35] P. Farail, P. Gaufillet, A. Canals, C. Le Camus, D. Sciamma, P. Michel, X. Cregut, M. Pantel, “The TOPCASED project: a Toolkit in Open source for Critical Aeronautic Systems Design”, ERTS, 2006.

[36] M.Y. Chkouri, A. Robert, M. Bozga, and J. Sifakis “Translating AADL into BIP - Application to the Verification of Real-Time Systems”, In MoDELS Workshops, pages 5-19, 2008.

[37] B. Berthomieu, F. Vernadat, Time Petri Nets Analysis with TINA, tool paper, In Proceedings of 3rd Int. Conf. on The Quantitative Evaluation of Systems (QEST 2006), IEEE Computer Society, 2006.

[38] N. Williams, B. Marre and P. Mouy, “Interleaving Static and Dynamic Analyses to Generate Path Tests for C Functions”, 3rd Workshop on System Testing and Validation (SV'04), December 2004, Paris, France

[39] ADeS, a simulator for AADL architectures, http://www.axlog.fr/aadl/ades_en.html

[40] MAST, Modeling and Anlysis Suite for Real-Time Applications, http://mast.unican.es/

[41] F. Singhoff, J. Legrand, L. Nana, L. Marcé, “Cheddar: a Flexible Real Time Scheduling Framework”, ACM SIGAda Ada Letters, volume 24, number 4, pages 1-8. Edited by ACM Press, New York, USA. December 2004, ISSN:1094-3641.

[42] H. Posadas, J. Angel Adámez, P.P. Sánchez, E. Villar, Francisco Blasco, “POSIX modeling in SystemC”, IEEE proc. Of the 11th Asia and South Pacific Design Automation Conference”, 2006.

[43] E. de las Heras, E. Villar, “Specification for systemc-AADL interoperability”, Workshop on Intelligent Solutions in Embedded Systems (WISES07), June 2007, Spain.

[44] A. Agarwal; S. Rajput, A.S. Pandya, “Power Management System for Embedded RTOS: An Object Oriented Approach”, Canadian Conference on Electrical and Computer Engineering, May 2006.

[45] Lab-sticc / CAT sourceforge, http://sourceforge.net/projects/lab-sticc/

[46] Open-PEOPLE : Open Power and Energy Optimization Platform and Estimator, http://www.open-people.fr/

FIGURES AND TABLES

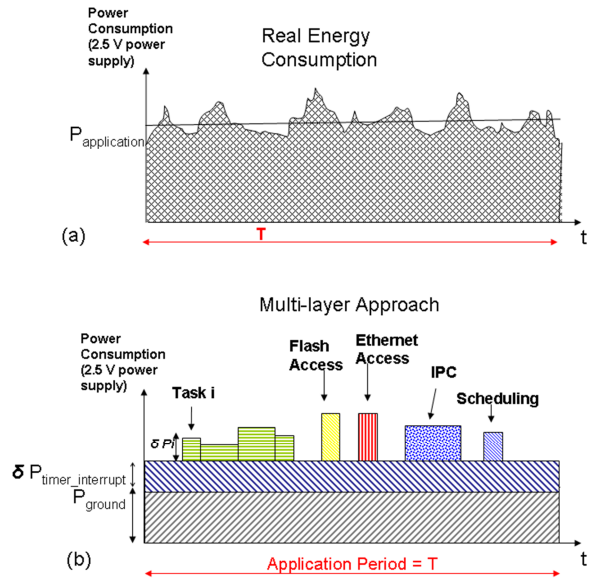


Figure 1. Multi-Layer energy estimation

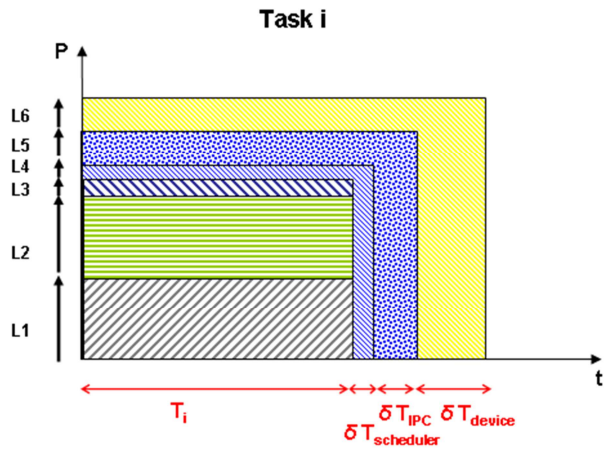


Figure 2. Multi-Layer energy estimation of individual task

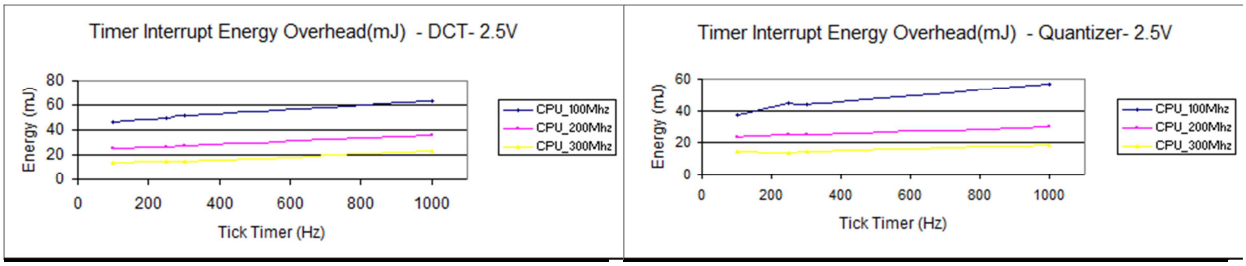


Figure 3. Timer energy overhead according to CPU and OS timer frequencies

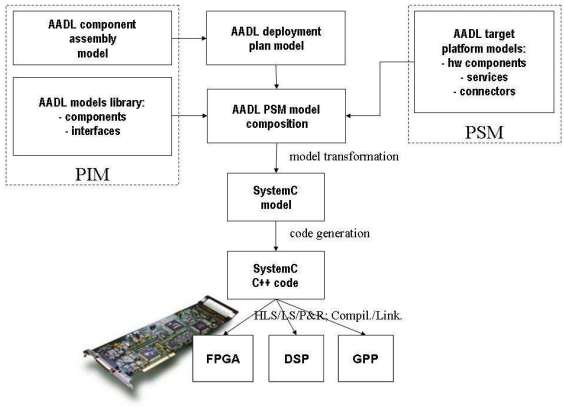


Figure 4. AADL component based design

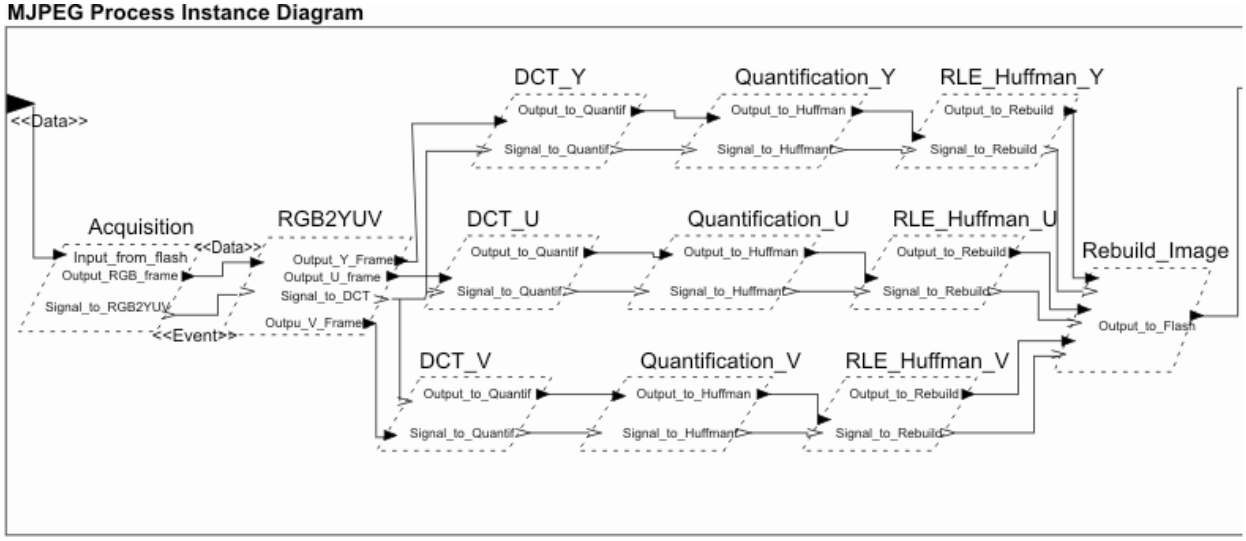


Figure 5. AADL model of MJPEG scenario 1

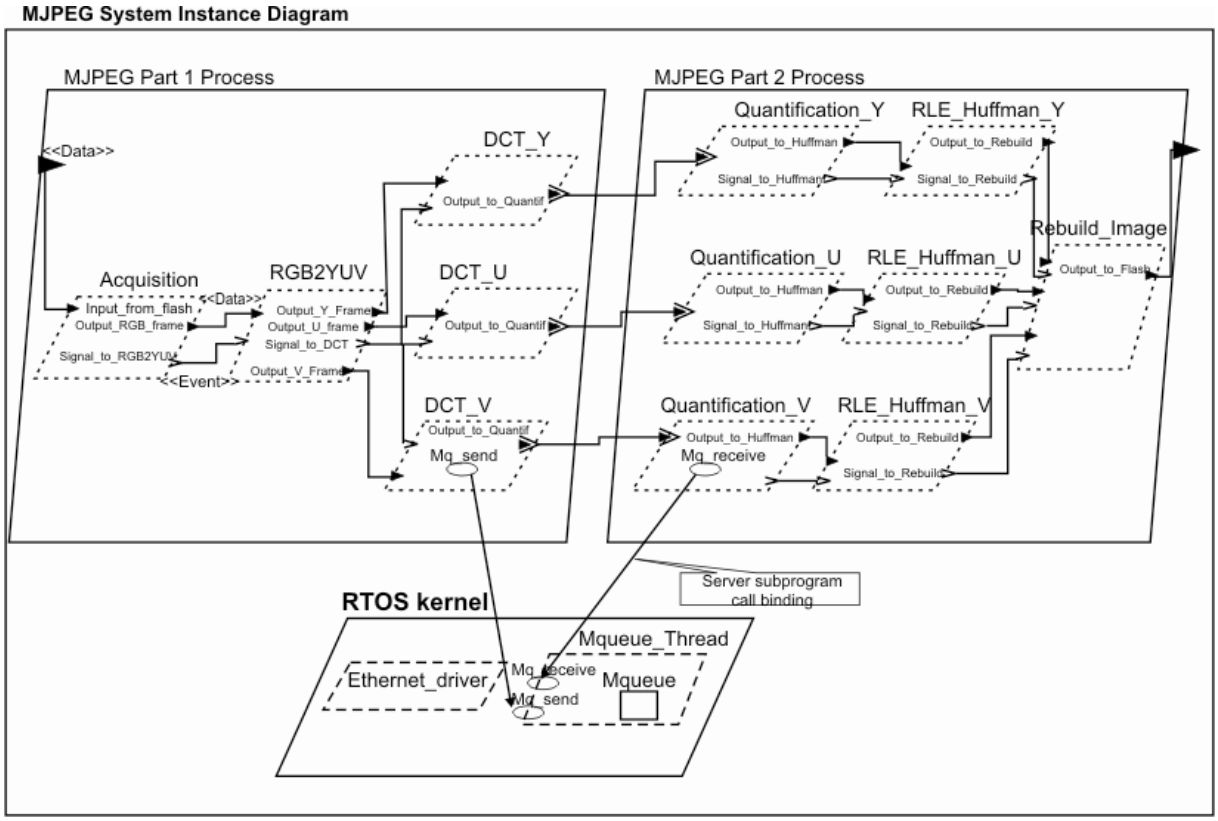


Figure 6. AADL model of MJPEG scenario 2

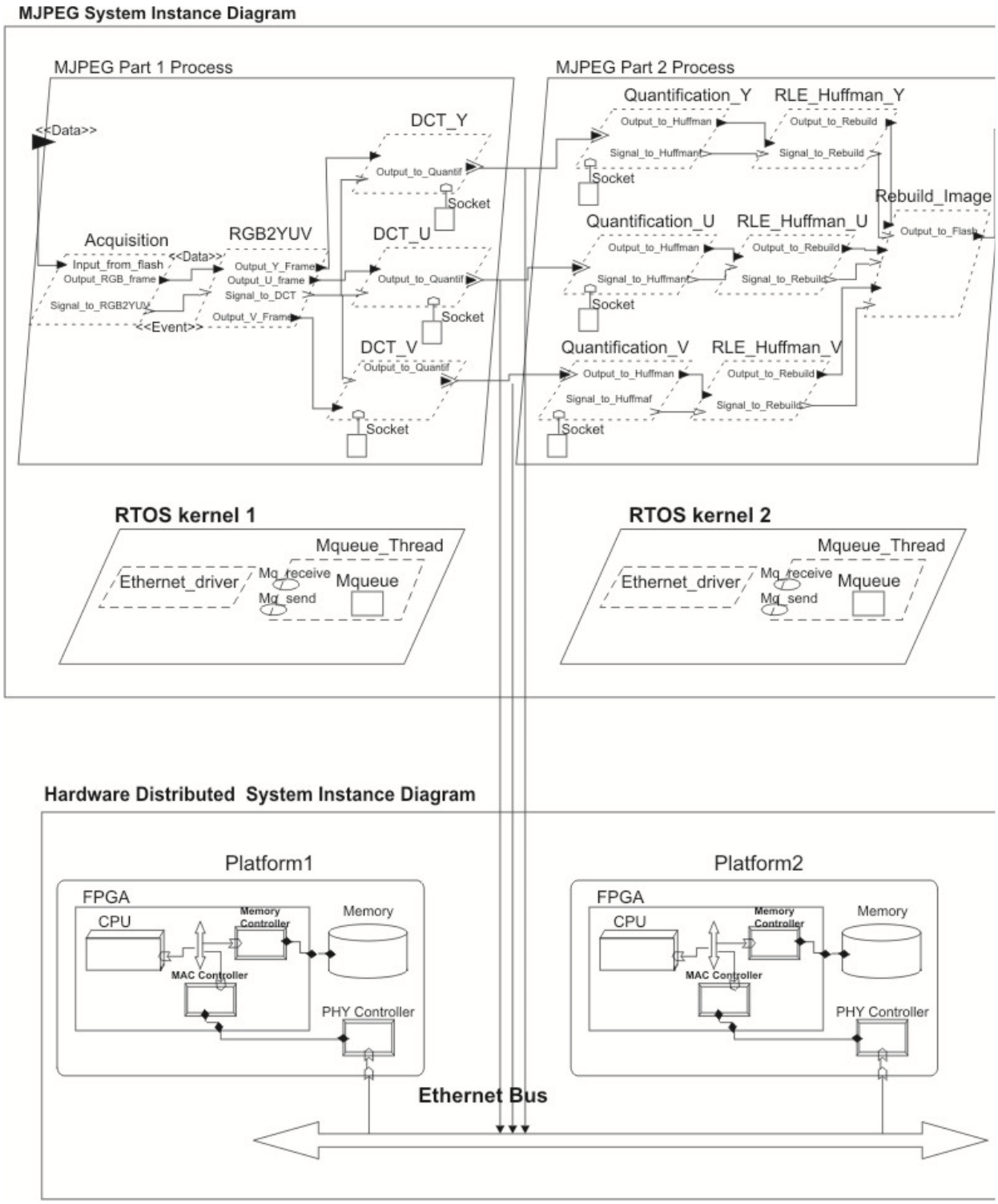


Figure 7. AADL model of MJPEG scenario 3

TABLE 1
POWER MODEL OF IPC

Hw Compo-nent	Power Model	Avg Error
SDRAM memory (2.5V)	$P_{mqueue}(mW) = 1{,}18F_{cpu} + 2341{,}9$	0,5 %
	$P_{mqueue}(mW) = 1{,}18F_{cpu} + 2341{,}9$	0,6 %
	$P_{mqueue}(mW) = 1{,}18F_{cpu} + 2341{,}9$	0,6 %

TABLE 2
MQUEUE ENERGY MODEL

CPU Frequency	$E(\mu J / Byte) = b \times Message_{size}{}^{p} + d$	**Avg Error**
SDRAM Memory (2.5V)		
Mq_send		
100Mhz	$350 \times X^{-0,99} + 0,02$	4,8%
200Mhz	$393 \times X^{-1} + 0,03$	2,3%
300Mhz	$511 \times X^{-0,99} + 0,04$	2,8%
Mq_receive		
100Mhz	$363 \times X^{-1} + 0,07$	2,2%
200Mhz	$405 \times X^{-0,99} + 0,09$	3%
300Mhz	$556,47 \times X^{-0,99} + 0,16$	1,6%

TABLE 3
MJPEG SCENARIOS

Scenario	**Tasks**	**Communication mechanism**
1	1 process, 12 threads, Periodic image acquisition	Threads synchronization with signals
2	2 processes, 12 threads, Periodic image acquisition	Process Communication through mqueue, Threads synchronization with signals
3	2 processes, 12 threads, Periodic image acquisition	Process Communication through sockets (Ethernet), Threads synchronization with signals

TABLE 4
ENERGY ESTIMATION OF MULTI-THREADED MJPEG TASKS (2.5V POWER SUPPLY)

τ_i	$\gamma(\tau_i)$	$P(\tau_i)$ (mW)	$T_i(ms)$	$E(\tau_i)$ (mJ)	δE_{timer} (mJ)	δE_{mqueue} (mJ)	$\delta E_{Ethernet}$ (mJ)	δE_{Flash} (mJ)
Acquisition	3,2%	2200	10,91	24	0,43	0	0	12,84
RGB2YUV	6,1%	2201	15,06	33,16	0,6	0	0	0
DCT_Y	1,3%	2199,42	29,83	65,61	1,19	5,39	29,82	0
Quantizer_Y	1,9%	2199,62	27,39	60,26	1,09	8,16	29,8	0
Huffman_Y	2,5%	2199,82	16,72	36,69	0,66	0	0	0
Rebuild_Image	3,7%	2200,21	5,8	12,77	0,23	0	0	4,23

TABLE 5
ENERGY ESTIMATION ON THE MJPEG SCENARIOS

Scenario	Measure(mJ)	Estimation(mJ)	Error
1	578,56	585,2	1,14%
2	600,95	625,85	4,14%
3 MJPEG part 1	651,16	719,88	10,55%
3 MJPEG part 2	653,4	728,47	11,48%

ANNEXE B

Annexe A

MemExplorer: From C code to Energy Optimal Memory Allocation

Johann Laurent, André Rossi & Marc Sevaux

Abstract *— In this paper, we propose to address the memory mapping problems in the software development context. Indeed, the software designers do not have both efficient methodology and tools that allow us to determine the optimum memory mapping for their applications directly from their C code program. Here, we propose mathematical model that is able to represent both all the memory conflicts generated by a given application and the memory constraints due to the memory hierarchy. Indeed in this paper, only the sub problem of optimum memory mapping for a given hierarchy is treated and not the whole problem of optimum mapping and hierarchy one. Our results, on classical digital signal processing applications, show that our methodology always generates the optimum mapping memory in reasonable time (less than 30 seconds) and that these mappings are always optimum in energy consumption point of view (energy improvement up to 75%). Our methodology is automatic since we have developed a too, MemExplorer, that generates the optimum mapping from C code and that this tool can be used by SoftExplorer (power/energy consumption estimation tool) so, this work is very user friendly for non expert engineers.*

Keywords — Memory mapping, Energy Consumption, C code, Integer linear program, Signal Processing applications

Introduction

Systems-On-Chip (SOC) are subject to real-time, surface and power consumption constraints. To meet these requirements, SOC designers use CAD tools enabling them to assess the impact of their choices early in the design process [3], [5]. As software layer complexity is growing in SOC, its impact on consumption is more and more significant, mostly because of the memory management of the data structures processed by the software layer. Indeed, in the case of image and signal processing applications, the number of data structures that have to be stored and/or loaded is very high (several mega/giga octets for HD broadcast for instance) [6], [8]. The method presented in [6] consists on code rewriting techniques that use loop and data-flow transformations with the aim of reducing the required amount of data transfers and storages while also improve access behavior. This consists in avoiding unnecessary copies of data, modifying computation order, shifting of “delay lines” through the algorithm to reduce the storage requirements, and re-computation issues to reduce the number of transfers and storage size. In [8], a dynamic trace of the application execution is realized in order to optimize the use of the Scratch Pad Memories (SPM) by the operating system.

Even if software applications do not process all the data at the same time, an important part of it has to be manipulated thus, stored in memory components. But despite technological improvements in access time, capacity and consumption, memory remains the bottleneck in many applications. Memory optimization is widely debated in the literature [1], [2], [4], [11], but most research effort is focused on the definition of an optimal memory hierarchy, memory segmentation and on data allocation (local and global variables) to a predefined memory hierarchy. In [1], an application profiling is realized with the aim of

analyzing the diverse memory segments used (stack, heap, code…). Then, these code and data segments are split into as many SPMs that‘s necessary (the number of SPMs depends on its size that is a priori fixed). In [2], the stacks are shared into several memory banks and/or memory hierarchy with the aim of reducing the access cost (access cost is defined by the necessary time to access to one data in the memory and its associated energy consumption) of local and global variables. In [4], the idea is to use SPMs to replace the cache memories. Indeed, the cache memories require more area than SPMs since these kinds of memories use tags to know if data are stored into the cache, so cache need more die area to store these tags; this area increase leads thus an energy consumption increase. Finally, [11] presents a methodology whose the aim is to reduce the SPM’ leakage energy consumption. Here, code transformations are realized so the data accesses are modified, these modifications lead a data sharing between several SPMs. Afterward, the temporal uses of these SPMs are studied in order to idle all the SPMs that are not used on a time window. As several SPMs are idled thus the energy consumption due to the memory part of the design is decreased.

They have been shown to bring significant improvements in access time and power/energy savings [6], [7], but their use is restricted to small and medium size chip design. When addressing memory management for more complex SOC, low level approaches become unusable because of its precise granularity and designers feel the need for higher level methods. Indeed, several SME (Small and Medium Enterprises) software engineers' interviews have shown that SOC designers often integrate some normalization committees' source code into their own application code.

These normalized codes are generally not optimized and practitioners do not have enough time (because of time-to-market pressure) and/or expertise to modify source code so as to optimize data placement for increasing performance.

For filling SOC designers' needs, a CAD tool should meet the following requirements:

- Keep the practitioners original design flow: the CAD tool must be integrated into a classical software flow.
- Fast optimization process. The optimized memory mapping must be returned in a reasonably short amount of time because of time-to-market pressure.
- User friendly. Practitioners have to spend as few times as possible learning to use the new tool, or modifying their code manually.

This paper presents a complete design flow that aims at meeting these requirements. To do so, complete data structures are manipulated as a whole instead of elementary memory words (i.e. scalar data). Data structures will be, for us, mono or multi dimensional arrays and/or pointers; these kinds of data are often used in signal and image processing applications. This approach granularity is obviously less precise, and performance improvements should also be fewer than with a low-level approach. However, the high-level proposed approach has the significant advantage of remaining practicable and efficient for large SOC design projects. Memory hierarchy is supposed to be fixed under the following assumption: one processor is able to use several memory banks (scratch pad) and no memory cache is available. Section 2 presents the design flow and the architecture on which the memory allocation process is used. This problem is modeled as an integer linear program in Section 3, and then computational results are provided for demonstrating the approach efficiency for different classes of applications (filtering, compression, communication, and

cryptography) are presented in Section 4. And finally, Section 5 concludes this paper and provides some prospects for future work.

The design flow

This section first introduces the target architecture used (i.e. the processor and the memory hierarchy), then the estimation flow (parameters extraction) will be presented and especially the determination of the memory conflict graph which will be allow us to compute the cost of each memory access by using the cost library.

Architecture used

Figure 50 presents the generic architecture used in the modeling scheme. This architecture is composed of one processor, on-chip memory banks (with $n \geq 0$) that can be accessed by the processor to load/store its data and one external memory that can also store data (typically the data that cannot be stored into the on-chip memories).

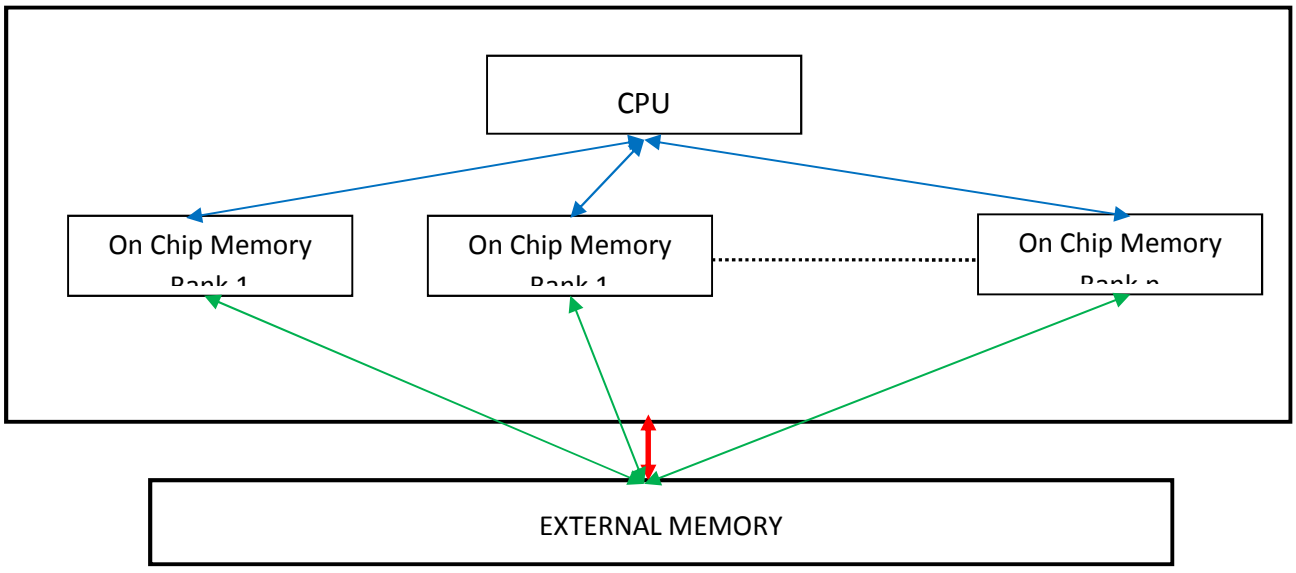


Figure 50: Generic Architecture

This type of memory architecture is often used in embedded systems because the scratch

pad memories have the advantage of consuming less power than cache memories. On the one hand the n on-chip memory banks are fast but small-sized memories, mostly a few KB. On the other hand, the external memory is far larger but also significantly slower. Indeed, the typical access time of this kind of memory is some milliseconds whereas on-chip memory access time is a few nanoseconds. These features play a key role in memory mapping: if the size of a data structure exceeds the on-chip memory banks capacity then it has to be allocated to the external memory. Furthermore, a data structure (for example one dimension array) whose size is less than on-chip memory bank capacity may be allocated to the external memory bank if the remaining on-chip memory bank capacity is too weak. Besides memory capacity, memory access time is also an important parameter: a data structure that is often accessed by the application should be placed in on-chip memory bank otherwise the whole application execution could be delayed.

These features (memory capacity and access time) are stored in a memory library for each memory type. This library enables MemExplorer users to get the best memory mapping fitting their architecture.

MemExplorer design flow

The first step in MemExplorer design flow is to extract two types of information from the application. The first one is the size (in byte) of all data structures used in the application and the second one is the cost of all conflicts involving the data structures. We remind of data structures can be, in our method, mono or multi-dimensional arrays and/or pointers. These extractions are performed automatically from the application source code. The MemExplorer Design Flow is shown on Figure 51.

The entry point of MemExplorer flow is a C source code provided by the user. This code is parsed by the modified front end of GCC 4.2. This parsing yields the size of each data

structure (pointers are also taken into account) and the conflict graph that points out which structures are accessed in parallel. These parallel accesses can generate access conflicts if the structures are mapped into the same on-chip memory bank or are both into the external memory (if these memories have one read/write port), or if one data structure is allocated an on-chip memory bank and the other one is allocated the external memory. The memory specifications are provided by the user because it depends on the memory used in his architecture. These features are stored into the memory library and used by MemExplorer in order to compute the conflict costs; so with this approach all the possible memory configurations (in size, number of banks, number of access ports) can be considered.

The library contains for each type of memory hierarchy the delay costs generate by the access conflicts. Of course, these costs depend on the memory type used (on chip or off chip, SRAM, SDRAM and so on...) and also depend on the processor used since the memory (on chip or off chip) access time differs from a processor to another one.

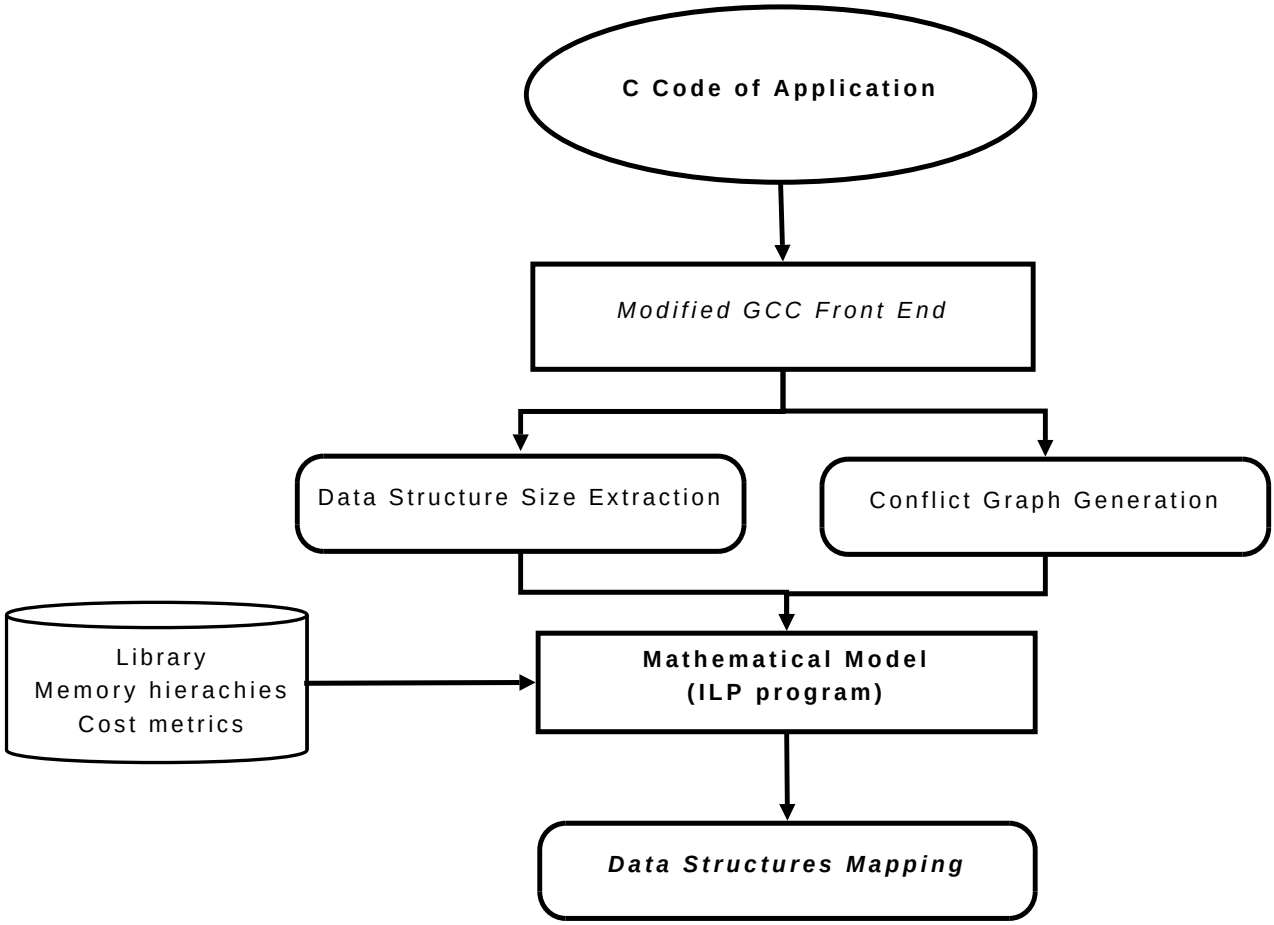


Figure 51: MemEplorer design flow

When the conflict graph and the data structure size extraction have been realized then, the mathematical model uses these information and the cost metric library (as presented in Figure 52) in order to generate the optimal data structure mapping.

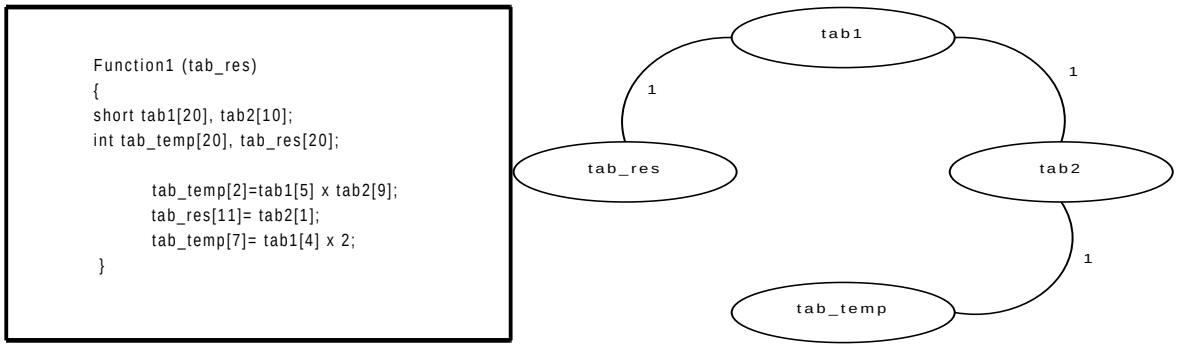


Figure 52: C Code and its associated memory conflict graph: Example 1

Figure 52 shows a brief application where only four data structures are used. These structures (which are arrays) are accessed in reading and/or writing mode.

The first step realized by the front end (developed from GCC 4.2) will be to extract the size of each data structure used in the application. This extraction is realized directly from the C code provided by the user. In the provided example (cf. Figure 3), we will have to extract the size of the four data structures: ***tab1***, ***tab2***, ***tab_res***, ***tab_temp***. The first two data in this application are structures of ***short***, where a ***short*** is assumed to be a two byte word. Thus, ***tab1*** has 20 elements of 2 bytes so its size is 40 bytes (the size of ***tab2*** is 20 bytes). ***tab_res*** and ***tab_temp*** have a size of 80 bytes all the elements of these structures is represented on 4 bytes; a ***int*** element being coded in 32 bits in this example (the ***int*** size depends on the data bus size of the processor used).

The second step consists in analyzing all the possible access conflicts (in writing and/or reading mode) between the different data structures: these conflicts are presented in the Memory Conflict Graph (MCG). Figure 3 shows the MCG for the first example.

In this Figure, we can see that ***tab1*** and ***tab2*** can be in conflict (if 2 accesses can be done in parallel on the target architecture) and that this conflict has a weight of 1. The weight of this first conflict corresponds to the memory accesses realized in the first expression of the provided example and we suppose that the load accesses are realized first. The weight carried by the arcs represents thus the maximum number of access conflicts. In similar manner, we can determine the weight of each arc of the Memory Conflict Graph. Finally, we can determine that ***tab1*** and ***tab_res*** are in conflict with a weight of 1 and that ***tab2*** and ***tab_temp*** are also in conflict with the same weight.

This first example shows a very simple case where no loops and/or control structure (if, else, switch...) are used. But now what's happen when control or loop structure are used by an application? In this case, we will study each LIB (Linear Instruction Block) separately and will define the memory conflicts by taking into account the iteration numbers (for the loops) and the probability to take each branch of control structures (for the if, else...). The second example (cf. Figure 4) shows a part of this rule application when there are control structures in the C code.

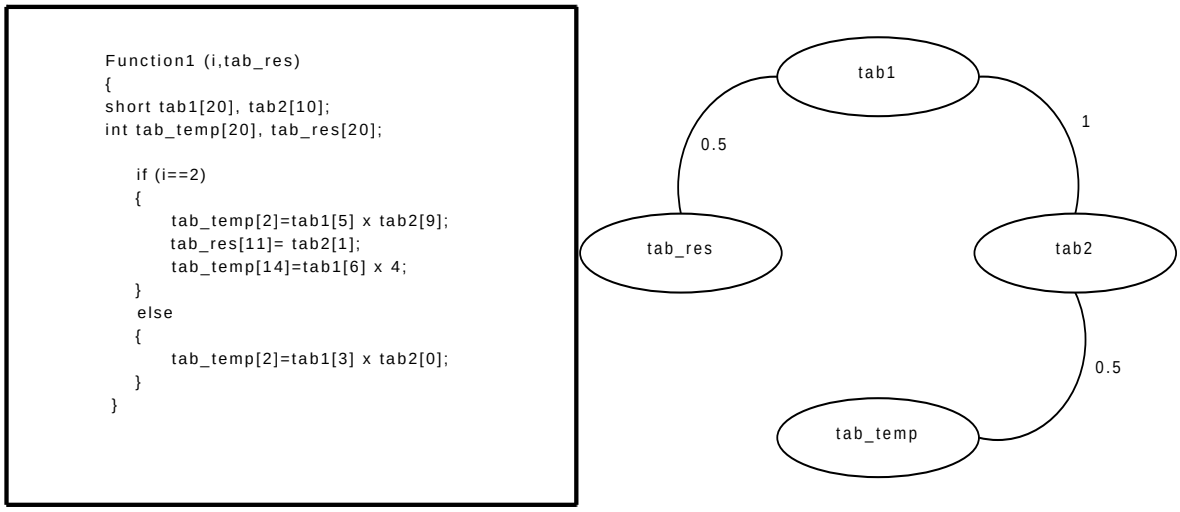


Figure 53: C Code and its associated memory conflict graph: Example 2

In the Figure 4, we can see that there are 3 possible conflicts: the first one between ***tab1*** and ***tab_res*** with a weight of 0.5, the second between ***tab2*** and ***tab_temp*** with also a weight of 0.5 and finally the last one between ***tab1*** and ***tab2*** with a weight of 1. This last conflict has a double weight compare to the others because one conflict exists between these two data in the IF branch but also in the ELSE one. So, as the probability to take each branch of the control structure is the same in this example (by default, the two branches are equiprobable), each conflict (in the IF and the ELSE branch) can occur one time with a

probability of 0.5. In conclusion, each conflict has a weight of 0.5 so the addition of these conflicts is carried by one arc of which the weight is 1. The MemExplorer user can change the execution probabilities of each control branch by using pragma inserted into his C code. These pragmas are used by our tool but do not taken into account during the compilation step; so the C code is not modified.

At last, the third example (cf. Figure 54) shows an application using loops and control structures.

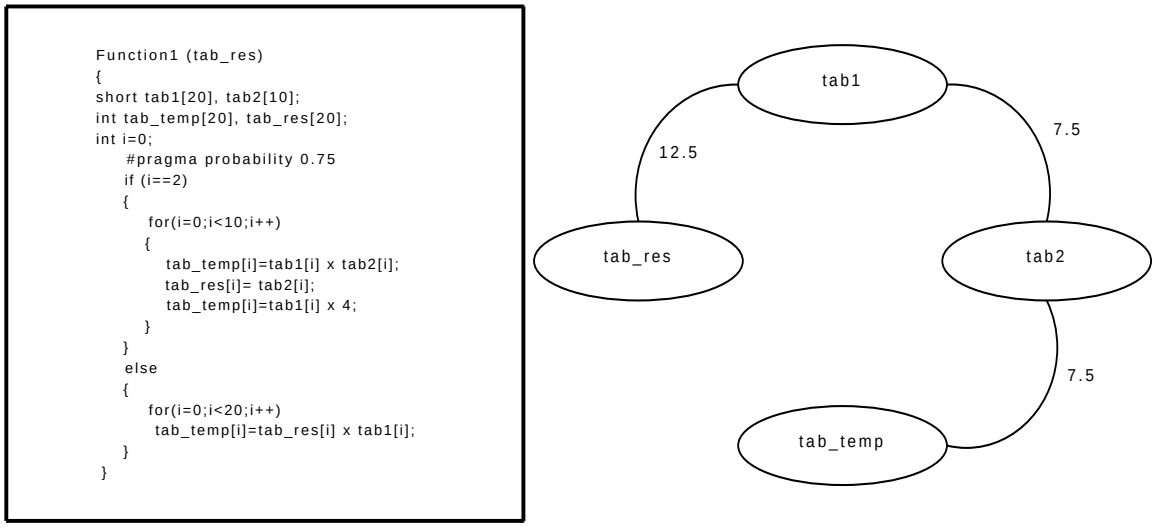


Figure 54: C Code and its associated memory conflict graph: Example 3

In the Figure 54, we have 3 memory conflicts: the first one between ***tab1*** and ***tab2*** with a weight of 7.5, the second between ***tab2*** and ***tab_temp*** with also a weight of 7.5 and at last the third conflict between ***tab1*** and ***tab_res*** with a weight of 12.5. Each weight is computed by using both the iteration number of loops, in which the conflicts are generated, and the probability to execute the branch control. In this example, the probability to execute the IF branch is equal to 0.75 (normalized to 1) since the user annotated his code thus, the probability to execute the ELSE branch is then equal to 0.25 (MemExplorer computes the ELSE probability by subtracting 0.75 from 1 so, the user does not need to annotate the ELSE).

Furthermore, the iteration number of the first loop is equal to 10 so the first conflict between ***tab1*** and ***tab2*** is computed as the following:

$$\text{Conflict Weight} = 0.75 \times 10 = 7.5 \quad (1)$$

One node of the graph (corresponding to data structures) cannot be linked to another one only by an unique arc then we must add the weight of similar conflicts generated into each LIB of the C code. As similar conflicts can exist in several C function of the whole application, we take into account the call number of each of these functions to weight the arc values of the MCG.

In complex applications, it can be difficult to determine the iteration numbers for the loops (mainly if these loops are dynamic), the probability to take a control branch and the number of function calls. So in this case to help the designer, MemExplorer can use a dynamic profiling of the application in order to automatically compute (the designer provides only the inputs for his application) the iteration numbers, the probability and the function calls.

Today, the first limitation of our flow is that we consider all the data structures as alive during the whole execution of the application. In fact, this is rarely exact since data can be used at the beginning of an application and never then. So with our methodology, we will obtain non optimal solutions since some memory areas would be able to be reused during the life time of the application. The second limitation is that we consider that if one data of the data structure (for example one data of an array) is in conflict with one or more data of another structure then this two data structure in their whole are in conflict. This is rarely happened in real applications, often only some parts of data structures are in conflict. So in

this case, we would have to split data structures into several smaller data structures and treat it singly. This division into several structures will lead to modify the C code of the application in order to manage the data transfers. Here, we do not want to modify the input C code because these modifications would have to be realized in an automatic way (it is not easy step) in order to avoid additional work for the designers.

Modeling the memory mapping optimization problem

This paper aims at mapping data structures to the memory banks and to an external memory so as to minimize to time spent loading these data. An integer linear program is designed for this purpose, using the notations introduced in the following subsections. Such approaches have been shown to be successful in Electronics [1] and also in Memory allocation problems [10], but the problem addressed here is quite different.

Problem data

The number of data structures is denoted n, and m is the number of available memory banks. The external memory is modeled as a fictive memory bank $m+1$. The size of data structure i is denoted s_i for all i in $[1,n]$ and the capacity of memory bank j is c_j for all j in $[1,m]$, the external memory has a capacity denoted c_{m+1}. Sizes and capacities are expressed in the same memory unit (Kbytes for instance). Besides its size, each data structure i is also characterized by an access cost denoted e_i for all i in $[1,n]$, that models the required amount of time to access data structure i provided it is mapped to a memory bank. If it is mapped to the external memory, its access cost is multiplied by a given integer p.

The number of conflicts between data structures is denoted o. Any conflict k in $[1,o]$ is modeled by the triplet (a_k,b_k,t_k), where a_k and b_k are two conflicting data structures, t_k being the cost of conflict k provided a_k and b_k are mapped to the same memory bank. Conflict costs and access costs are expressed in the same time unit (milliseconds for instance). If a_k or

b_k is mapped to the external memory, the conflict cost is multiplied by p. More formally, conflict k has four different statuses:

- Status 1: a_k and b_k are mapped in two different memory banks. The conflict does not generate cost.
- Status 2: a_k and b_k are mapped in the same memory bank. The conflict generates the cost t_k.
- Status 3: a_k and b_k are such that one of these data structures is mapped in a memory bank and the other one is mapped in the external memory. The conflict generates the cost $p.t_k$.
- Status 4: a_k and b_k are mapped in the external memory. The conflict generates the cost $2.p.t_k$.

It must be stressed that a data structure can be conflicting with itself. This case arises when two subparts of the same data structure are involved in the same instruction in the original C code, as in ***a[i] = a[i+1] + b[i]*** . All these data are assumed to be provided by the user. m, and c_j for all j in $[1,m+1]$ and p describe the architecture, n, s_i and e_i for all i in $[1,n]$ describe the data structures, whereas the conflicts list (a_k,b_k,t_k) for all k in $[1,o]$ carries information on both architecture and data structures.

Problem variables

Problem variables are used to model the solution, i.e. the mapping of data structures to the memory banks and to the external memory. They are also involved in the objective formulation, as well as in the problem constraints' statement. The numerical value of these variables is set by solving the mixed integer linear program shown in section 3.4. Boolean

variables $x_{i,j}$ are used to model the mapping itself: for all i in $[1,n]$ and for all j in $[1,m]$, $x_{i,j}$ is set to one if data structure i is mapped to memory bank j, it is set to zero otherwise. Analogously, $x_{i,m+1}$ is set to one if data structure i is mapped to the external memory, it is set to zero otherwise. As conflicts have a significant impact on the solution cost, real nonnegative variables are used to model their statuses. More formally, y_k is equal to the cost of conflict k according its current status i.e. y_k has four possible values: 0, t_k, $p.t_k$ or $2.p.t_k$. The solution description is complete when each problem variable is assigned a numerical value.

Problem objective and constraints

The present section focuses on the problem objective and constraints formulation. The solution cost (i.e. the mapping cost) is the total time spent accessing the data structures and gathering them in the appropriate registers to perform the required operations listed in the C file. The problem aims at founding the mapping that minimizes this cost. It can be written as follows:

$$Minimize \quad \sum_{k=1}^{o} y_k + \sum_{i=1}^{n} \sum_{j=1}^{m} (e_i \times x_{i,j}) + p \sum_{i=1}^{n} (e_i \times x_{i,m+1}) \qquad (2)$$

The solution cost is the sum of three terms. The first one is the total cost generated by the conflicts. The second term is the access cost of all data structures mapped to a memory bank whereas the last term is the access cost of all data structures mapped to the external memory. Besides, the following constraints have to be satisfied to yield a valid mapping.

First, any data structure has to be allocated either a unique memory bank or the external memory:

$$\sum_{j=1}^{m+1} x_{i,j} = 1 \quad \forall \; i \in [1, n] \qquad (3)$$

Second, the total size of the data structures allocated to a memory bank (or to the external memory) must not exceed its capacity:

$$\sum_{i=1}^{n} x_{i,j} \times s_i \leq c_j \forall j \in [1, m+1] \qquad (4)$$

Third, for any conflict $\boldsymbol{k}$, variable $\boldsymbol{y_k}$ must be set appropriately. This is enforced through the four following constraints:

$$\begin{aligned}
&x_{a_k,i} + x_{b_k,j} \leq 1 + \frac{1}{t_k} y_k \quad \forall j \in [1, m], \forall k \in [1, o] \\
&x_{a_k,j} + x_{b_k,m+1} \leq 1 + \frac{1}{p \times t_k} y_k \quad \forall j \in [1, m], \forall k \in [1, o] \qquad (5) \\
&x_{a_k,m} + x_{b_k,j} \leq 1 + \frac{1}{p \times t_k} y_k \quad \forall j \in [1, m], \forall k \in [1, o] \\
&x_{a_k,m+1} + x_{b_k,m} \leq 1 + \frac{1}{2 \times p \times t_k} y_k \quad \forall k \in [1, o]
\end{aligned}$$

The first inequality prevents $\boldsymbol{y_k}$ from being less than $\boldsymbol{t_k}$ if conflict $\boldsymbol{k}$ has status 2. The next two inequalities prevent $\boldsymbol{y_k}$ from being less than $\boldsymbol{p.t_k}$ if conflict $\boldsymbol{k}$ has status 3, and the last one prevents $\boldsymbol{y_k}$ from being less than $\boldsymbol{2.p.t_k}$ if conflict $\boldsymbol{k}$ has status 4. Since the problem is a minimization one, any optimal solution is such that $\boldsymbol{y_k}$=0 (respectively $\boldsymbol{y_k=t_k}$, $\boldsymbol{y_k=p.t_k}$ ***and*** $\boldsymbol{y_k=2.p.t_k}$) if conflict $\boldsymbol{k}$ has status 1 (respectively status 2, status 3 and status 4).

Mixed integer linear program

The problem is stated as a mixed integer linear program by gathering the above problem objective and constraints (see below):

In the next section, this mixed integer linear program is used for a TI C6201 with different types of algorithms. Two software tools are compared: the mixed integer linear program is first modeled with Xpress-Mosel ® language version 2.0.0 and solved with the Xpress-Optimizer ® version 19.00.00 [9]. Second, it is modeled and solved with GLPK 4.9, a GNU

package [10]. Both these solvers implement branch-and-cut approaches that are state-of-the-art techniques for addressing integer linear programming problems [15].

$$
\begin{aligned}
&\textit{Minimize} \quad \sum_{k=1}^{o} y_k + \sum_{i=1}^{n} \sum_{j=1}^{m} (e_i \times x_{i,j}) + p \sum_{i=1}^{n} (e_i \times x_{i,m+1}) && \\
&\sum_{j=1}^{m+1} x_{i,j} = 1 && \forall\ i \in [1, n] \\
&\sum_{i=1}^{n} x_{i,j} \times s_i \leq c_j && \forall\ j \in [1, m+1] \\
&x_{a_k,i} + x_{b_k,j} \leq 1 + \frac{1}{t_k} y_k && \forall\ j \in [1, m], \forall\ k \in [1, o] \\
&x_{a_k,j} + x_{b_k,m+1} \leq 1 + \frac{1}{p \times t_k} y_k && \forall\ j \in [1, m], \forall\ k \in [1, o] \qquad (6) \\
&x_{a_k,m} + x_{b_k,j} \leq 1 + \frac{1}{p \times t_k} y_k && \forall\ j \in [1, m], \forall\ k \in [1, o] \\
&x_{a_k,m+1} + x_{b_k,m} \leq 1 + \frac{1}{2 \times p \times t_k} y_k && \forall\ k \in [1, o] \\
&x_{i,j} \in \{0,1\} \quad \forall\ i \in [1, n], && \forall\ j \in [1, m+1] \\
&y_k \geq 0 && \forall\ k \in [1, o]
\end{aligned}
$$

Computational results

For our experiments, we used a Digital Signal Processor board based on the TI TMS320C6201 processor. This processor is a VLIW processor which enables to execute up to 8 instructions in parallel. Among these 8 instructions, two of them can be load and/or store instructions so the processor can access twice in memory in the same time but for that, these accesses must be done in the internal memory and furthermore in two separate banks. Even if, the processor can directly access the on-chip memory banks or the external one, these two memories have not the same characteristics. Indeed first, the access time of these memories is very different since the on-chip bank can be accessed in one CPU cycle whereas the external memory uses 16 CPU cycles (if the processor is clocked at its maximum frequency). These access times allow us to compute the conflict costs that are used in order to minimize our cost function. The Second, their power consumption are also different that will imply various energy consumptions. Finally, the last difference is their size since each memory bank has 32KBytes whereas the external memory has 8 MBytes (in our EVM board). The size of

memories allows us to know during the mapping step if there is still memory space to map data structures into the memory banks.

We used classical Signal Processing application to prove that our approach is an interesting way to improve application's global performances. Most of these applications is come from the UTDSP test suite:

- Adpcm application implements an adaptive differential pulse coded modulation encoder that adheres to the CCITT G.172 standard for 32 Kbits/s ADPCM speech coding.

- Compress application uses the discrete cosine transform to compress a 128 x 128 pixel image by a factor of 4:1 while preserving its information content.

- Dwt application uses the discrete wavelet transform to compress a 512 x 512 pixel image.

- Edge application detects the edges in a 256 gray-level 128 x 128 pixel image. The program relies on a 2D-convolution routine to convolve the image with kernels (Sobel operators) that expose horizontal and vertical edge information.

- Fir_1024 application implements a classical finite impulse response filter using 1024 samples.

- G721 uses the Marcus Lee algorithm to perform a G721 coder.

- Histo enhances a 256-gray-level, 128x128 pixel image by applying global histogram equalization.

- LMS realizes a LMS filter for 2 audio channels.

- LPC implements a linear predictive coding encoder.
- Enc_mpeg2 performs a MPEG-2 encoder.
- Dec_mpeg realizes a MPEG-1 decoder.
- Spectral calculates the power spectral estimate of an input sample of speech using periodogram averaging.
- Treillis implements a soft-decision sequential decoding algorithm for a (32, 16) extended BCH code.

Instance Name	Optimal cost	Running time with Xpress-Optimizer	Running time with GLPK
Adpcm	224	0.031s	0.015s
Compress	511232	0.047s	0.015s
Dwt	3.16e+6	0.046s	0.406s
Edge	2.70e+6	0.047s	0.002s
Fir_1024	0	0.016s	0.002s
G721	0	0.031s	0.002s
GSM	86132	0.015s	0.015s
Histo	1.26e+6	0.031s	0.015s
LMS	2046	0.031s	0.031s
LPC	790	0.062s	0.031s
Enc_mpeg2	0	24s	1.172s
Dec_mpeg	786.5	0.328s	0.140s
Spectral	640	0.063s	0.002s
Treillis	12.06	0.187s	3.140s

Table 1: Comparing the computation time with Xpress-Optimizer and GLPK

The proposed mapping, while focused on time minimization, turns out to be energy-saving compared to the mapping produced by basic compilers.

Table 2 presents results for several classical digital signal processing applications obtained by using the SoftExplorer estimation tool [12]. Table 2 compares energy consumption and execution time ***(Texe)*** for the proposed mapping and for the basic compilers' mapping. Of course, the execution time of the ILP depends on the computer used, but also on the solver. The results suggest that execution time remains low for many applications. Increasing the instance size and the density of the conflict graph may lead to longer execution times. More precisely, the problem is a combination of a bin packing problem (because the memory bank capacity is limited) and of the vertex coloring problem (because of the conflict graph). As it is well-known that both these NP-hard problems can be very difficult or very easy on two instances of the same size, the number of data structures and the number of memory banks are not the only reasons for the problem difficulty. We have used the SoftExporer tool [12] in order to obtain the results on classical digital signal processing applications. SoftExplorer uses Energy/Power models obtained from physical measures and these models have a maximal error of 5%. In the Softexplorer power models, only the CPU and on-chip memory are taken into account on the other hand the off chip power consumption is estimate since different kinds of external memory can be used (SBSRAM, SDRAM …). For the energy consumption, the delay due to the external memory access time is however taken into account (SoftExplorer assumes that the external memory plugged is appropriate).

These results show how our memory mapping, using our mathematical modeling approach, improves both the execution time and the energy consumption of these applications. We can see that the execution time can be improved up to 89% for LMS filter application and less than 1% for Spectral. This is due to the fact that Spectral application uses an algorithm where

few memory accesses can be done in parallel so few memory conflicts are generated when this algorithm is executed thus our optimization algorithm does not realized many improvements.

Table 2 also shows that the energy improvement is always lower than the execution time one. Indeed for instance, if we analyze the results in energy point of view for the mpeg2 encoder application (Enc_mpeg2), we can see that the execution time improvement is equal to 81% whereas the energy improvement is only equal to 64.5%. This is due to the fact that when we optimize the memory mapping, we decrease the memory conflict number so we can increase the potential number of memory accesses that could be done in parallel. Thus, if more memory accesses are realized in parallel the power consumption induced will be greater so, the benefit obtained on the execution time will be partially lost by the fact that the power consumption will be increased.

Instance Name	Texe (basic mapping)	Texe (our mapping)	Gain (%)	Energy (basic mapping)	Energy (our mapping)	Gain (%)
Adpcm	0.171ms	0.159ms	7	0.893mJ	0.868mJ	2.8
Compress	14.4ms	13.89ms	3.55	42.46mJ	41.56mJ	2.1
Dwt	315ms	63.35ms	80	773mJ	240mJ	69
Edge	50.9ms	16.2ms	68	162mJ	89mJ	45
Fir_1024	26.22ms	21ms	20	102mJ	91mJ	11
G721	0.46ms	0.45ms	2.2	3.46mJ	3.45mJ	0.3
GSM	72.4ms	70.8ms	2.2	415mJ	411mJ	1
Histo	0.6ms	0.51ms	15	2.01mJ	1.84mJ	8.45
Lms	461ms	52ms	89	1156mJ	291mJ	75
Lpc	2.28ms	2.26ms	0.9	10.6mJ	10.5mJ	0.9
Enc_mpeg2	5760ms	1103ms	81	15277mJ	5433mJ	64.5

Dec_mpeg	0.101ms	0.070ms	30.7	0.463mJ	0.360mJ	22
Spectral	53.7ms	53.6ms	0.19	184.7mJ	184.6mJ	0.05
Treillis	0.054ms	0.053ms	1.85	0.238mJ	0.236mJ	0.85

Table 2: Energy consumption and execution time results

Conclusion

We have presented here a whole design flow to optimize the memory mapping a data structure from C code application. This design flow can be integrated into any software design flow since it is completely automated so the user has nothing to do just providing the C code of its own applications. MemExplorer will use the code to generate first, the data conflict graph and then will use this information as input for its mathematical model to compute the optimal memory mapping. Finally, MemExplorer will provide to the software designer a mapping file that he will use to realize the linking step into its own framework. Our methodology and its associated tool allow improving the execution time up to 89% and the energy consumption up to 75% and even if the improvement brought can be, in some case insignificant, our method never decreases the performances.

Today, we generate a static memory mapping and we do not take into account that some data are used only during a part of the application's execution and that this memory space could be reused since this data will not always be read or written by the application. Take into account this new parameter allows us to improve the memory mapping optimization since in this case, we could map more data structures into low level memory (memory bank) so this would increase the execution time of the whole application but would generate additional memory traffic between upper and lower memories. Another improvement will be to take into account that, often, only subparts of data structures are in conflict and not the full structures. In this case, some improvements will be realized in comparison to our approach but it will be

necessary to model more precisely what subparts of data structures are in conflict.

REFERENCES

[1] Ahmad and Chen, “Post-Processor for Data Path Synthesis Using Multiport Memories”, IEEE/ACM International Conference on Computer-Aided Design, ICCAD-91, November 11-14, 1991, Santa Clara, CA, USA, pp. 276-279.

[2] F. Angiolini, L. Benini, and A. Caprara, “An efficient profile-based algorithm for scratchpad memory partitioning”, Processing of the IEEE Transaction on CAD of Integrated Circuits and Systems, (June 2005), pp. 1660-1676.

[3] O. Avissar, R. Barua, and D. Stewart, “An optimal memory allocation scheme for scratch pad based embedded systems” ACM Translation of embedded Computer Systems (2002), Vol. 15, pp.6-26.

[4] F. Balasa, P. G. Kjeldsberg, A. Vandecapelle, M. Palkovic, Q. Hu, H. Zhu and F. Catthoor, “Storage estimation and design space exploration methodologies for the memory management of signal processing applications” Journal of Processing Systems (2008), Vol. 14, pp.51-71.

[5] R. Banakar, S. Steinke, B. S. Lee, M. Balakrishnan, and P. Marwedel, “Scratchpad memory: design alternative for cache on-chip memory in embedded systems” Processing of of the 10^{th} international symposium on hardware/software codesign (2002), pp.73-78.

[6] L. Benini, and G. Micheli, “System Level Power optimization: techniques and tools” ACM Translation of Design Automation Electronic Systems (2000), Vol. 12, pp.115-192.

[7] F. Cathoor, K. Danckaert, S. Wuytack, and N. Dutt, “Code transformation for data transfert and storage exploration preprocessing in multimedia processors” IEEE Design and Test (2001), Vol. 18, pp.70-82.

[8] A. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. Brodersen, “Optimizing power using transformation” IEEE Transaction on Computer Aided Design of integrated Circuits and Systems, Vol. 14, pp.12-31.

[9] D. Cho, S. Pasricha, I. Issenin, N. Dutt, M. Ahn, and Y. Paek “Adaptative scratch pad memory management for dynamic behavior of multimedia application ” IEEE Transaction on Computer Aided Design of integrated Circuits and Systems (2009), Vol. 28, pp.554-567.

[10] G. De Micheli, “Synthesis and Optimization of Digital Circuits,” MsGraw-Hill, 1994.

[11] FICO, Xpress-MP (2009), http://www.dashoptimization.com.

[12] GNU, GLPK linear programming kit, (2009), http://www.gnu.org/software/glpk.

[13] M. Kandemir, M.J Irwin, G. Chen, and I. Kolcu, “Compiler guided leakage optimization for banked scratch-pad memories” IEEE Transaction on Very Large Scale Integration Systems (2005), Vol. 28, pp.1136-1146.

[14] J. Laurent, E. Senn, N. Julien, and E. Martin, “SoftExplorer: estimating and optimizing the power and energy consumption of a C program for DSP applications” Eurasip Journal on Applied Signal Processing (2005), Vol. 16, pp.2641-2654.

[15] L. Wolsey, “Integer programming”, Wiley-interscience, 1998

ANNEXE C

High-level Interconnect Delay and Power Estimation

Antoine Courtay[1,2,*], Johann Laurent[1], Nathalie Julien[1] and Olivier Sentieys[2]

[1]LESTER, University of South Brittany, Lorient, 56100, France first_name.last_name@univ-ubs.fr

[2]IRISA, University of Rennes1, Lannion, 22300, France last_name@irisa.fr

It is now well known that interconnects represent a bottleneck for delay, power consumption and area on chips. To face these problems, some works have been realized around performance optimizations. However, results presented in this paper, show that optimization techniques do not always face good criteria for interconnect performance optimizations. We therefore have developed a high-level estimation tool based on transistor-level characterizations, which provides fast and accurate results for time and power consumption estimation. Estimation results allowed us to determine a new interconnect consumption model and also enabled to find some new key issues that have to be pointed out for future performance optimizations.

Keywords: interconnects, power consumption, timing, modeling, estimation, performance optimization.

1. INTRODUCTION

Today, nomad applications are more and more complex and require many computation resources, implying a strong volume of data to be stored or to be translated. To transfer these data from memories to computation units or from computation unit to another one, we have to use interconnect networks. Interconnect power consumption and area can represent up to 50% of the overall consumption as well as chip area.[1,2] Indeed, the transistor and wire dimension evolution results in a modification of the circuit behavior, mostly of the propagation time; the propagation time for a wire becomes higher than in a gate[2]. A part of this increase is due to the evolution of wire resistance. Indeed, between two technologies, dimensions of wire decrease and thus the resistance, which is inversely proportional to the wire section, increases. Moreover with the increase of interconnect network density and of metal layers number, the part of the chip dedicated to interconnects is increasing, so the interconnect power consumption part compared to the overall chip power consumption is increasing.

In older technologies (higher than 250nm), interconnects were made up of aluminum and were separated from each other by an insulator whose permittivity bordered 4. With the evolution of the manufacturing processes[3], aluminum has often been replaced by copper whose conductivity is smaller. With this evolution, propagation time on interconnects has strongly decreased.

Simultaneously with this replacement of aluminum by copper, insulators with weak permittivity appeared. The advantage of using these insulators is to reduce the crosstalk phenomena and wire-to-substrate capacitances. Despite of these technological evolutions, interconnects are, now, known to be the bottleneck of the system. Thus, it is essential to take interconnect power consumption and delay into account during the design phase of a system.

In this paper, we propose, after the presentation of power consumption model of buses, a new estimation tool that allows the user to obtain different kinds of results about the power consumption of his interconnect networks. Furthermore, we propose a new transition classification from the power consumption point of view; this classification is very different than the previous one defined from a propagation time point of view. Then, based on this new classification and on other statistical metrics, we propose some new ways to optimize the interconnect performances (delay, power consumption).

This paper is organized as follows. Section 2 presents physical parameters and power performance models for wire and bus. Crosstalk effects and model are carefully presented. The estimation flow as well as our estimation tool is introduced in Section 3. In Section 4, our tool is used to analyze criteria used by performance optimization techniques; some new ways for optimization are also discussed. Last section concludes this paper.

2. CHARACTERIZATION FLOW: FROM WIRE TO BUS

The first step for interconnect modeling is to represent as realistic as possible interconnect behavior. In order to obtain the best precision in time and power consumption, experiments must be realized at physical level. Therefore we decided to model interconnects with a transistor-level characterization using a SPICE simulator.

Firstly, this section introduces physical modeling from wire to the complete bus system with all parasitic phenomena.

2.1. The wire

Physical parameters which allow wire modeling are:

- *R*, wire resistance, expressed in Ohm [Ω];
- *C*, wire capacitance, expressed in Farad [F];
- *L*, wire inductance, expressed in Henry [H].

Their variation depends on the wire characteristics (metal composition, wiring level) as well as dimensions. Inductance has an impact only for very deep submicron technologies (underneath 45nm) and for extremely long wires.[4] Furthermore, it is an important phenomenon when peaks of current or strong voltage variations occur (typically the clock tree and the power lines); so inductance will not be taken into account in our model dedicated for interconnect buses. Thus we have chosen to consider only *RC* model for the wire.

It is possible to characterize the wire with elementary parameters which can be found in manufacturer Design Kits. These parameters are:

- $R_{sq} = \rho / T$, resistance by square (also called resistance per sheet), expressed in Ohm by square [Ω/*square*] with ρ the metal resistivity and *T* the wire thickness;
- C_{sq} , wire lower face to substrate elementary capacitance, expressed in Farad per meter [F/m];
- C_e , wire edge to substrate elementary capacitance, expressed in Farad per meter [F/m].

Using these three parameters, it is possible to compute the wire resistance and capacitance according to its dimensions (see Fig. 1.): length (L) and width (W) expressed in meter [m]. Note that C_{sq} and C_e depend on the height (H) between the wire and the substrate and thus depend on the metal level used.

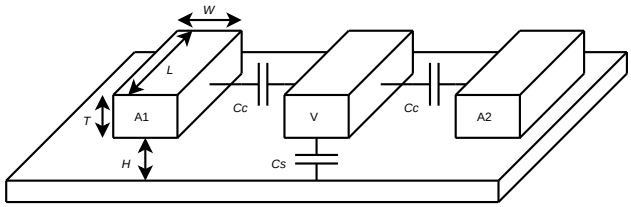


Fig. 1. A victim wire (V) and its two agressors (A1, A2). C_s represents the total wire to substrate capacitance as expressed in Equation (4) and C_c the crosstalk capacitance as expressed in Equation (5) according to physical dimensions of the wire: length *L*, width *W*, thickness *T* and height *H*

The total resistance of the wire is given by the following equation:

$$R = R_{sq} \frac{L}{W} \tag{1}$$

The total capacitance of the wire is, actually, the sum of two capacitances: the total wire lower face to substrate capacitance (parallel-plate capacitance) noted C_{pp} , and the total wire sides to substrate capacitance (fringing capacitance) noted C_f .

$$C_{pp} = C_{sq} WL \tag{2}$$

$$C_f = 2C_e L \tag{3}$$

Factor 2 in equation (3) is intended to include the fact that the two sides of the wire contribute to C_f .

The total wire capacitance to substrate can be expressed by:

$$C_s = L\left[C_{sq}W + 2C_e\right] \quad (4)$$

Now, we discuss about the distribution of *R* and *C* on the wire in order to model its behavior as accurate as possible. The lumped model is a simple interconnect model; it consists in putting end-to-end values of *R* and *C* found previously. However, its precision is much less reliable in terms of propagation time than a model where *R* and *C* are distributed. In this manner, *R* and *C* values can be split indefinitely. For the $\Pi 3$ model, which consists in distributing the wire resistance on three resistances and the wire capacitance on four capacitances, the values obtained in terms of time are close to experimental values. We have retained $\Pi 3$ model for our experiments because of its simplicity and its precision (estimation error of time less than 5%). [5]

2.2. The bus

We consider a *n* bit bus which consists of *n* parallel wires of the same length and at least *2n* buffers (*n* input buffers, *n* output buffers and maybe others if bufferization is used) and allows propagating data between two cells. Using several wires reveals a new capacitive coupling which is the coupling between wires. The coupling capacitance between two adjacent wires, known as crosstalk capacitance C_c , depends on the area facing each other; so it depends on the following dimensions: wire thickness (*T*), wire length (*L*) and wire spacing (*S*).

$$C_c = \varepsilon_0 \frac{TL}{S} \text{ where } \varepsilon_0 \text{ represents } SiO_2 \text{ permittivity} \quad (5)$$

When transitions occur on adjacent wires, there is a generation of an unwanted noise due to the coupling capacitance. The noise due to the crosstalk is relatively localized. In general, a system with crosstalk is modeled by neglecting higher-order effects on non-adjacent wires; thus, we only consider the effect on three wires as represented in Fig. 1. The coupling capacitance between wires is also distributed on the nodes of the distributed $\Pi 3$ *RC* model defined previously as represented in Fig. 2. We explain in more details the crosstalk phenomenon and associated effects in the next paragraphs.

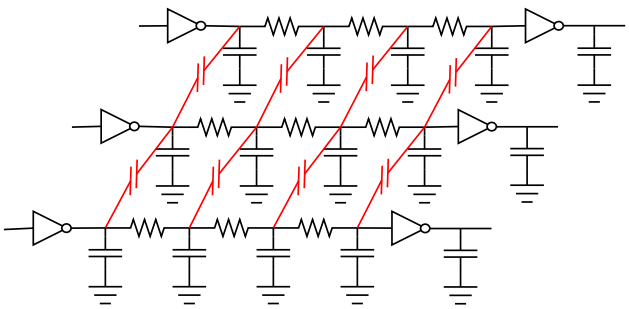


Fig. 2. A 3-wire $\Pi 3$ bus with crosstalk capacitance (in red) distributed on the nodes of the *RC* $\Pi 3$ wire model. In a circuit, buses are connected to some other gates after output buffers, that is the reason why a capacitance whose size is equivalent to the input capacitance of a minimal size gate (ie: an inverter of size one) must be added

Effects, due to crosstalk, can be summarized into three categories.

- The first one is that crosstalk induces noise; indeed the coupling capacitance between adjacent wires introduces a permanent link between them. When a transition occurs on a wire (aggressor), its neighbors (victim) are affected because a voltage peak is generated on them.[6] There are two categories of coupling, positive and negative coupling respectively when the amplitude of the noise exceeds a positive or a negative voltage value on the victim. Noise peaks above *GND* or under V_{dd} are the most tedious because they can cause errors if their values are greater than the buffer threshold voltage at the end of the bus (cf. Fig. 3. a). With technology shrinking, noise due to crosstalk compared to the overall noise increases since the coupling capacitance also rises between two technology steps. As a result, the voltage peak generated by the coupling capacitance is more and more important, compared to the voltage swing on the bus.
- A second issue is the increase in propagation time. When the victim and its aggressor(s) are switching simultaneously, a voltage peak is generated. This peak can, according to the configuration of transitions, slightly accelerate (in the case of simultaneous transitions in the same direction) or slow down (in the case of simultaneous transitions in opposite side) the propagation on the victim wire (cf. Fig. 3. b). A transition classification has been carried out according to propagation time on the victim: this classification is presented in Table I. *g* represents the delay factor and *r* the ratio of crosstalk capacitance compared to wire capacitance to substrate. Here, ↑ represents a rising transition, ↓ represents a falling one and - means that there is no transition on the wire. In the best case, when wires are switching in the same direction, the delay is the delay without crosstalk (i.e. *g*=1). However, data transmission on the bus must be clocked regarding the worst propagation time case (i.e. *g*=1+4*r*). Considering a real case, where $C_c = C_s$, the propagation time can be multiplied by five or more.[5]
- Finally, the last issue is an increase in power consumption. Indeed, power consumption depends linearly of the capacitance presented by a device. Since the wire capacitance C_{eff} depends on the crosstalk capacitance value (cf. Table I), crosstalk contributes to increase dynamic power consumption.[8]

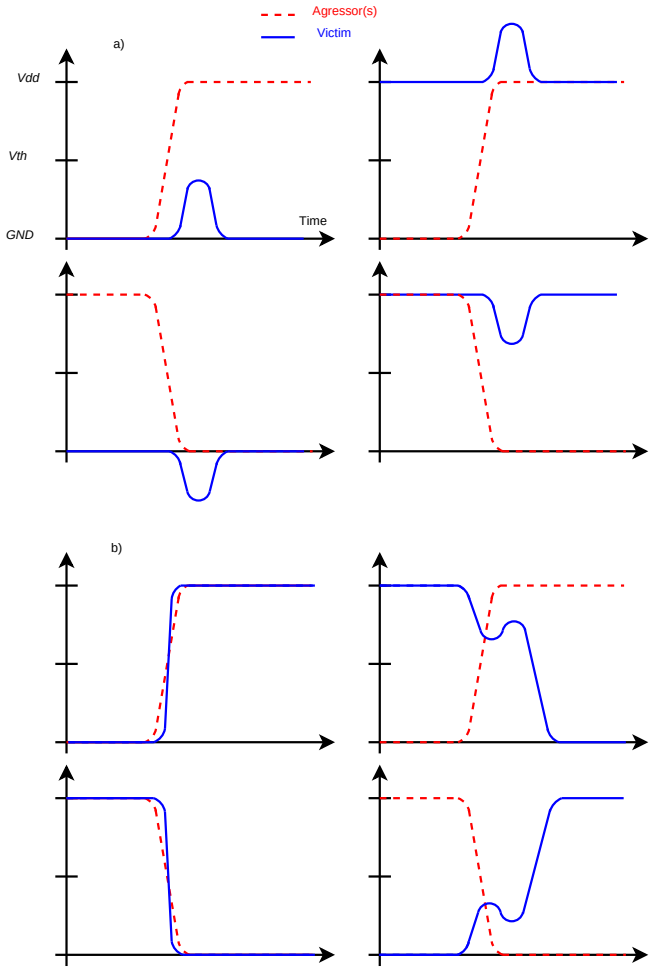


Fig. 3. Errors and timing due to crosstalk. a) the victim remains on a stable level when aggressors (dashed lines) switch. b) victim and aggressors switch at the same time.

Table I. Effective capacitance C_{eff} and delay factor *g* of the victim wire and corresponding transition patterns. *g* represents the delay factor and *r* the ratio of crosstalk capacitance compared to wire capacitance to substrate. Here, ↑ represents a rising transition, ↓ represents a falling one and - means that there is no transition on the wire. Transition patterns are represented as follows: (transition on the aggressor wire 1, transition on the victim wire, transition on the aggressor wire 2)

C_{eff}	Transition patterns		g
C_s	(↑, ↑, ↑)	(↓, ↓, ↓)	1
$C_s + C_c$	(-, ↑, ↑)	(-, ↓, ↓)	$1+r$
	(↑, ↑, -)	(↓, ↓, -)	
$C_s + 2C_c$	(-, ↑, -)	(-, ↓, -)	$1+2r$
	(↓, ↓, ↑)	(↑, ↓, ↓)	
	(↑, ↑, ↓)	(↓, ↑, ↑)	
$C_s + 3C_c$	(-, ↑, ↓)	(-, ↓, ↑)	$1+3r$
	(↑, ↓, -)	(↓, ↑, -)	
$C_s + 4C_c$	(↑, ↓, ↑)	(↓, ↑, ↓)	$1+4r$

The last parameters that have to be defined for bus modeling are resistance, input and output capacitances of the buffers involved in the bus. The number of buffers depends on bus length and on buffering technique used. These parameters can be easily found by using transistor dimensions and parasitic parameters provided in technology libraries and by using formulas described in [5].

Previously, we have seen that the evolution of the resistive and capacitive parameters introduces delay over the data propagation time. Thus when this delay becomes too critical, especially for long interconnects, designers have to use some buffer insertion methods in order to accelerate the data propagation. Many works have already been realized on buffer insertion for interconnects in [10-12]. These are based on the formula of the propagation time introduced in [13]. They aim at founding an optimal value in terms of buffer number (K_{opt}) and strength (H_{opt}) in order to obtain the best temporal performance.

Knowing physical parameters that compose the entire bus, we are able to model propagation time and power consumption. The first step of the modeling process is to identify which parameters impact significantly delay and power consumption.

- The first parameter is the technology and its associated number of metal layers. Each metal layer has its own physical characteristics (dimensions) and usage: the lowest metal layers are used for very short wires (inside cells), intermediate layers for buses (between cells), and finally the highest layers are used for clock tree and power lines.

- The second parameter is the metal layer used in the considered technology. It is well known that wire resistance and capacitance vary with the metal layer, since dimensions (thickness, spacing, high and width) are different with the layer used.
- The third parameter is the wire length since this parameter impacts capacitance and resistance. When interconnects are quite long or when time is critical, it is necessary to insert repeaters along wires, thus, repeated and non repeated lines have to be modeled.
- Crosstalk capacitances have effects on power consumption and propagation time, as shown in Table I, thus the different kinds of transitions will be also parameters for modeling. As said before, the crosstalk capacitances are split on the wire by using the $\Pi 3$ model.

Using these parameters, power consumption and delay modeling can be realized at the circuit level using SPICE simulations (we used ELDO v5.7 in this paper). These simulations have been accomplished for three different technologies (130, 90 and 65nm). The results obtained with SPICE, in terms of time and energy consumption, have been summarized in multi-input tables for the different parameters. These tables are used by the high-level estimation tool that will be presented in the next section.

3. ESTIMATION FLOW

3.1. Estimation flow presentation

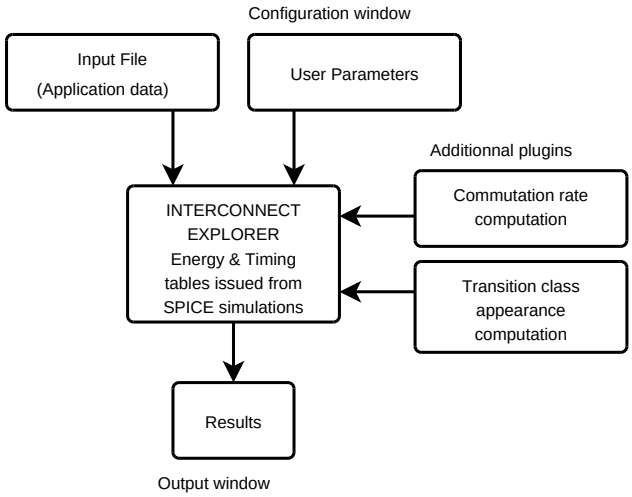


Fig. 4. Estimation flow used for bus computation results

A tool, called Interconnect Explorer, has been developed for high-level estimation of interconnect performances. This tool is based on the energy and timing transistor-level characterization result tables (SPICE simulations). The estimation flow used by Interconnect Explorer is explained in Fig. 4 and detailed hereafter.

When using Interconnect Explorer, users have to choose their bus configuration by setting the following parameters in the tool configuration window (Fig. 5.): technology, metal layer, bus length,

bus width, frequency and bufferization type. Users have also to provide an input file; this file contains the application data that are crossing over the bus.

Some additional plugins have been included in this tool to compute commutation rate per bit on the bus and also the probability of appearance of each transition class (defined in Table I). Commutation rates per bit are obtained by using the data input file. We compute the activity on each wire by realizing the ratio between the number of transitions on the wire compared to the total number of data.

By the same way, the probability of appearance of each transition class is obtained by computing the ratio between the occurrence number of each transition class compared to the total number of transition occurrences.

When configuration is done, Interconnect Explorer provides to the users, in the output window (Fig. 6.), results in terms of:

- energy consumption,
- static power consumption,
- average dynamic power consumption,
- maximum dynamic power consumption,
- instantaneous dynamic power consumption,
- maximum frequency allowed on bus (determined by worst case transition),
- area on the bus (area for wires and buffers),
- commutation rate per bit (useful to evaluate performance optimization techniques),
- and percentage of appearance of each transition class of Table I (same remark as above).

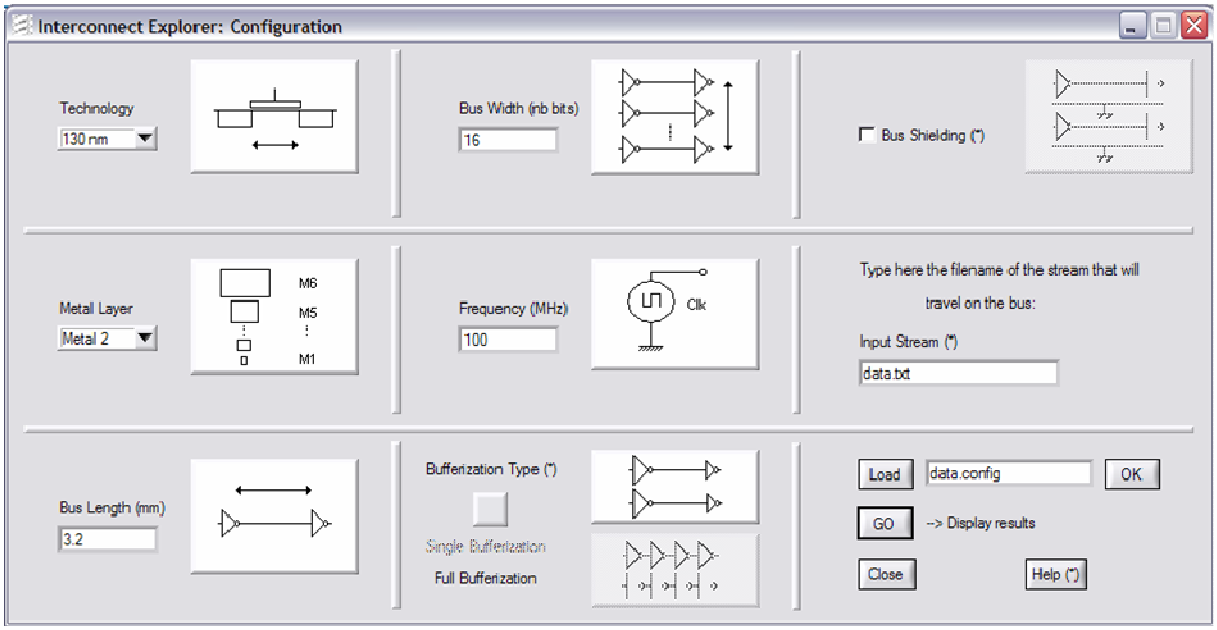


Fig. 5. Interconnect Explorer configuration window screenshot

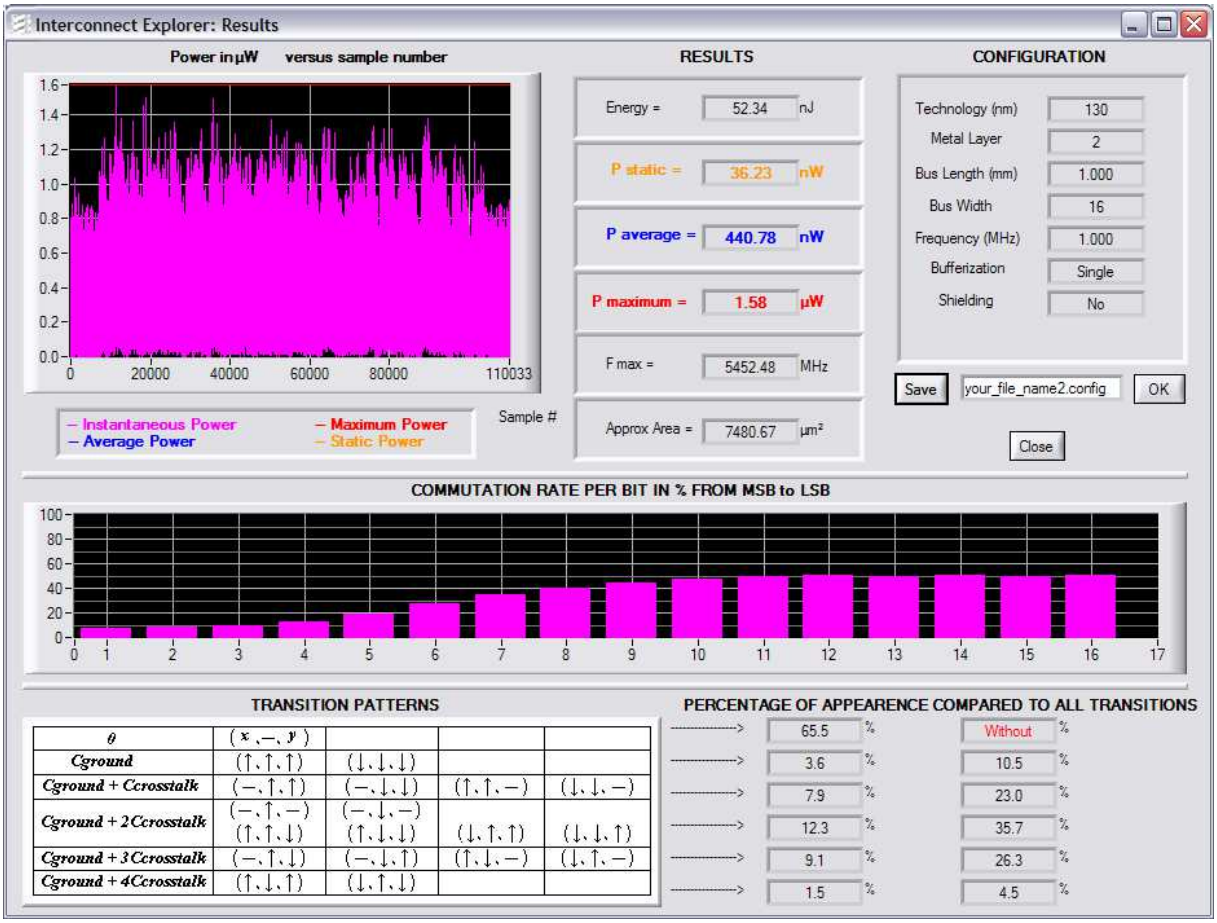


Fig. 6. Interconnect Explorer results window screenshot

In next paragraph, Interconnect Explorer is used in order to check the transition table classification presented in Table I.

3.2. Transition table validity

In the state of the art on performance optimizations for interconnect, most of the proposed techniques realize the elimination of most tedious transition classes of Table I (i.e. 1+3*r* and 1+4*r*). For instance, we can cite shielding [14-15], skewing [16] and temporal coding [17] techniques. As these techniques assume that the transitions that are the most tedious for delay are the same for consumption, we checked these results using Interconnect Explorer.

Experiments have been carried out using our tool on metal layers reserved for buses with a length of 1mm in a 65nm technology (let us note that results are the same for other technologies and metal layers). Table II shows propagation time and power consumption for different transition patterns of bus data. Results are given for the simple bufferization case of Fig. 7.a. Results of Table II (transitions

are classified from the weakest value to the strongest) show that the temporal transition classification, according to the importance of the capacitance seen by the victim wire, is the same that the one presented in Table I. In a second time, it is important to note that the transition classification, from a power consumption point of view, is not similar to the delay classification. Table II, for its power consumption section, can be divided into two parts:

- In the high part of the table, transitions are exclusively rising and are classified from those having the lowest capacitance to those having the strongest.
- In the low part of the table, transitions are exclusively falling and are classified in the same manner.

It is very important to note that the power consumption used by rising transitions is always similar, whereas for falling ones the power consumption increases with the growth of the capacitance.

Table II. Delay and power consumption results according to Table I transition patterns. Delay are measured as computing the time taken by the output to switch half the supply voltage value and is given in pico-seconds. Power consumption is given as energy per transition expressed in femto-joules.

Delay classification		
(-, -, -)	0	0 ps
(↑, ↑, ↑)	C_s	49 ps
(↓, ↓, ↓)	C_s	49 ps
(-, ↑, ↑)	$C_s + C_c$	67 ps
(-, ↓, ↓)	$C_s + C_c$	67 ps
(↑, ↑, ↓)	$C_s + 2C_c$	99 ps
(-, ↑, -)	$C_s + 2C_c$	99 ps
(-, ↓, -)	$C_s + 2C_c$	99 ps
(↓, ↓, ↑)	$C_s + 2C_c$	99 ps
(-, ↑, ↓)	$C_s + 3C_c$	139 ps
(-, ↓, ↑)	$C_s + 3C_c$	139 ps
(↓, ↑, ↓)	$C_s + 4C_c$	173 ps
(↑, ↓, ↑)	$C_s + 4C_c$	173 ps

Energy classification		
(-, -, -)	0	0.21 fJ
(↑, ↑, ↑)	C_s	13.29 fJ
(-, ↑, ↑)	$C_s + C_c$	13.43 fJ
(-, ↑, -)	$C_s + 2C_c$	13.45 fJ
(↑, ↑, ↓)	$C_s + 2C_c$	13.89 fJ
(-, ↑, ↓)	$C_s + 3C_c$	14.10 fJ
(↓, ↑, ↓)	$C_s + 4C_c$	14.86 fJ
(↓, ↓, ↓)	C_s	33.77 fJ
(-, ↓, ↓)	$C_s + C_c$	92.00 fJ
(-, ↓, -)	$C_s + 2C_c$	150.35 fJ
(↓, ↓, ↑)	$C_s + 2C_c$	150.73 fJ
(-, ↓, ↑)	$C_s + 3C_c$	207.76 fJ
(↑, ↓, ↑)	$C_s + 4C_c$	265.07 fJ

In order to understand why falling transitions consume more than rising ones, it is necessary to know when the current is extracted from the power supply. Two cases can be considered. Interconnect lines can be simply bufferized (one input buffer and one at the end of the line, Fig. 7.a) or full bufferized (Fig. 7.b) according to the desired performance. Depending on transition type (rising or falling), line capacitance (or line segments in a full bufferization case) are charged or not, as illustrated by Fig. 7.

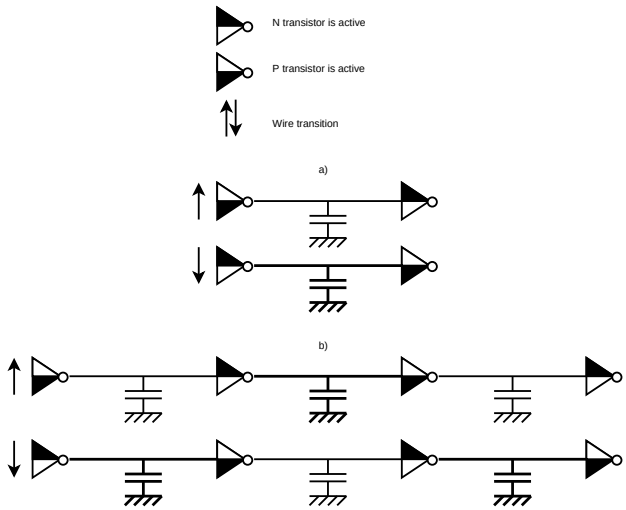


Fig. 7. Line or segments charged (in bold) according to bufferization state. The high or low part of the inverter symbol is blackened depending on the inverter transistor (*PMOS* for high and *NMOS* for low) activated according to wire transition type

- For simple bufferization (cf. Fig. 7.a), when a rising transition occurs the *NMOS* transistor is activated, and thus the line capacitance is not charged through the power supply. In the other case, when a falling transition occurs, it is the *PMOS* transistor which is activated, and thus the line capacitance is charged by the current coming from the power supply.
- For full bufferization (cf. Fig. 7.b), buffers which are inserted on the line must be even so that signal at the output of the line is the same as at the input. Thus, there is always an additional segment to be charged with falling transition compared to rising one. Consequently, falling transitions are more penalizing in terms of power consumption than rising ones.

To sum up, experiments allow us to conclude that temporal worst cases transitions are not equal to consumption worst case transitions, since falling transitions consume more energy than rising ones (all classified according to the importance of the capacitance presented by the line).

In the rising case, power consumption varies from only 5.6% around the average value, and all transitions can be classified in the same category. In fact, power consumption for rising transition is due to shortcut path between the power supply and the ground during output switching. Table III presents the consumption transition pattern classification.

Table III. Transition patterns classification for energy consumption where *j* is the transition pattern type and is varying from 0 to 4

$C_{L(i,j)}$	Transition patterns	Consumption	*j*
X	(?, ↑, ?)	$E_{shortcut}$	X
C_s	(↓, ↓, ↓)	$E_{i,j} = E_{i,0}$	0
$C_s + C_c$	(-, ↓, ↓) , (↓, ↓, -)	$E_{i,j} = E_{i,1}$	1
$C_s + 2C_c$	(-, ↓, -) , (↓, ↓, ↑) , (↑, ↓, ↓)	$E_{i,j} = E_{i,2}$	2
$C_s + 3C_c$	(-, ↓, ↑) , (↓, ↑, -)	$E_{i,j} = E_{i,3}$	3
$C_s + 4C_c$	(↑, ↓, ↑)	$E_{i,j} = E_{i,4}$	4

With this new transition pattern classification, a more accurate dynamic power consumption model is defined in this section. The energy consumed on an N_{bit} –bit bus is defined as follows:

$$E_{dynamic} = \sum_{i=0}^{Nbit-1} \sum_{j=0}^{4} \left[P_{i,j}.E_{i,j} \right] \quad (6)$$

where $P_{i,j}$ is the probability of having a *j* type transition pattern on wire *i* and $E_{i,j}$ is the corresponding energy consumption. *j* is varying from 0 to 4 as shown in Table III and *i* from 0 to $N_{bit} - 1$.

For a full transition cycle (if there is a rising transition on a wire there is of course a falling one after), the energy consumption $E_{i,j}$ can be computed by the following equation:

$$E_{i,j} = E_{shortcut} + C_{L(i,j)}.V_{dd}.V_{swing} \quad (7)$$

where $E_{shortcut}$ is due to shortcut path between the power supply and the ground during output switching for a rising transition and the $C_{L(i,j)}.V_{dd}.V_{swing}$ term represents the energy computation due to charging of load capacitance and to shorcut. In this term V_{dd} represents the supply voltage, V_{swing} the switching voltage and $C_{L(i,j)}$ is the loading capacitance of a *j* type transition pattern on wire *i* which can be computed as follows, $C_{L(i,j)} = C_s + j.C_c$ as shown in Table III with $j \in [0,4]$. The dynamic power consumption can be computed as follows:

$$P_{dynamic} = E_{dynamic}.F \quad (8)$$

where *F* is the frequency on the bus for data sending. By substituting Equation (6) in Equation (8), we obtain for $P_{dynamic}$:

$$P_{dynamic} = \sum_{i=0}^{Nbit-1} \sum_{j=0}^{4} \left[P_{i,j}.E_{i,j}.F \right] \quad (9)$$

$$P_{dynamic} = \sum_{i=0}^{Nbit-1} \sum_{j=0}^{4} \left[P_{i,j}.F.\left(E_{shortcut} + C_{L(i,j)}.V_{dd}.V_{swing} \right) \right] \quad (10)$$

4. ANALYSIS OF PERFORMANCE OPTIMIZATION TECHNIQUES

In this section, a non exhaustive state of the art of performance optimization techniques at different abstraction levels is discussed. Then, their performance on key parameters impacting timing and consumption are analysed by using Interconnect Explorer.

4.1. Technological level

4.1.1. Wire sizing

Since it is known that wire capacitances are dependant to technological dimensions, a first method consists in the modification of wire dimensions:

- Height (*H*) and width (*W*) to reduce the wire capacitance to substrate (C_s);
- Thickness (*T*) and spacing (*S*) to reduce the crosstalk capacitance (C_c).

The method suggested in [18] consists in modifying spacing between wires in a non uniform way; i.e. a wire *i* and its neighbour $i+1$ are separated by a space of S_1 and the wire $i+1$ and its neighbour $i+2$ are separated by a space of S_2 and so on. Results show a reduction of bus power consumption up to 30% as well as a decrease in propagation time since crosstalk capacitance is decreased. Drawbacks are an increase of bus area since spacing between wires is increased.

4.1.2. Spatial shielding

Shielding consists in inserting additional wires between bus wires. These additional wires can have fixed logical levels or levels that change with data transmitted. The first case is called static shielding and the second dynamic shielding.

The first type of static shielding consists in inserting between each signal of the bus a wire connected to the ground or to the supply voltage. This type of shielding makes the elimination of all cross transitions (such as type $1+3r$ and $1+4r$) possible. So, there is a strong acceleration of the data transmission since only $1+2r$ transitions remain. On the other hand, transitions of type 1 and $1+r$

are eliminated. Indeed each victim wire which carries out a transition, is surrounded by wire whose level does not change; this causes to defer less consuming transitions in a category with more consuming transitions. Data activity remains unchanged.

An evolution of this technique can be found in [14] where the shielding technique consists in doing an alternative shielding to the ground and to the supply voltage. Performance in terms of speed and power consumption are the same as the previous technique. One advantage of this technique is to, in addition to shielding, distribute supply lines through the chip.

In [15], the selected shielding technique is the following: the shield wire has the logical level of the logic *AND* of its two neighbours. Since the shield wire level moves with each data transmission, this shielding is called dynamic shielding.

Another very simple method of shielding consists in duplicating each signal by transmitting on the shield wire the same signal as its neighbour. The acceleration brought by this technique is higher than the one presented in [14], because the case where the two aggressors are stable is eliminated.

In conclusion, the main advantage of shielding techniques is that they considerably increase data transmission on the bus since the worst cases of Table I are eliminated. On the other hand, they are not efficient in term of area since the number of wires is doubled and have a limited impact on energy.

4.2. Logical/circuit level

4.2.1. Signal skewing

The solution presented in [16] consists in intentionally shifting the signals to avoid having simultaneous transitions on two neighbour wires. Even and odd wires are shifted temporally, thus a wire (that is even or odd) changes when its two neighbours are stable. In this manner the worst case of transition will be $1+2r$. The acceleration brought by this technique is very limited due to the fact that the data transmission is slowed down. Indeed, latency is added between the transmission of the even and odd bits. Simulation results show that acceleration can lie between 5 and 20%. This technique is based only on simulations and needs a complex design for the transmitter and receiver clock; moreover there is no implementation proposed.

4.2.2. Dynamic voltage scaling

Authors of [19] propose to use several values of the supply voltage for the buffers in order to limit voltage excursions on the lines. The principle of the method is to realize dynamic adaptation of the supply voltage (Dynamic Voltage Scaling: DVS) of these repeaters according to the operation frequency which is imposed to them. Simulation results show an average reduction of 4.6x on dynamic power together with a 15.2% latency reduction. This technique implies the addition of several analogical control blocks who aim at controlling the voltage switching.

4.3. Architectural level

The majority of the optimization techniques have been proposed at the architectural level. The *n* wires of the bus are coded into *m* bits with $m \geq n$ such as the coded data activity is lower than that the original one. The various techniques of data coding are either optimized for the address or the data bus.

4.3.1. Gray code (address bus)

The idea of this coding technique presented in [20-21] is to have only one transition on the bus each time the acceded addresses are consecutive. This coding is called Gray code or reflected binary code. The experiments carried out in [21] claim a 33% activity reduction and a 77% power consumption reduction on the bus. The more the bus is large and the more decoding is long; indeed in the decoder, gates are cascaded from MSB to LSB, which implies a long critical path.

4.3.2. T0 code (address bus)

The suggested idea in [22] is to add a wire noted *INC* that will be switched to 1 when the reached addresses are consecutive. This technique reduces the activity to 0 when the reached addresses are consecutive, which strongly reduces consumption on the bus. Unfortunately the design of coder and decoder is rather complex (register benches, adders, multiplexers…) and the codec over cost consumption is greater than the bus consumption reduction for wire lengths in today's SOC (System On Chip). An evolution of this technique can be found in [23] where several increment steps are defined for consecutive accesses.

4.3.3. Bus Invert / Partial Bus Invert (data bus)

The idea of the coding technique presented in [24] is to compare the number of bits changing between data $n-1$ at clock cycle $t-1$ and data n at clock cycle t. If this difference (Hamming distance) is higher than half of the bus width, then the data n sent at cycle t is inverted. It is necessary to send to the decoder an additional line noted *INV* to invert or not data received. This technique, called Bus Invert, is efficient for large buses where data activity is quite strong. Indeed, if Bus Invert is applied to the whole bus, its use will likely be less frequent because the *MSB* are often correlated. Therefore in [25] Bus Invert is only applied to the part of the bus which has the strongest activity; this coding is called Partial Bus Invert.

4.3.4. Code book (data bus)

The aim of the Code book coding technique presented in [26] is to store *i* old values transmitted on the bus and to transmit to the current cycle the value which has less Hamming distance with those transmitted to the i previous cycles. It is then also necessary to send the code to the decoder on 2^i additional wires. Results show a decrease in bus consumption, but for extremely long wires (more than 7.5*cm*), which can not be found in nowadays Systems on Chip.

4.3.5. Temporal Shielding (data bus)

The temporal shielding presented in [17] consists in forcing the bus down to 0 between each transmitted data. In this manner, there is no cross transition left and the worst case is here $1+2r$. On the other hand, power consumption increases because it is necessary to transmit twice more data of the case without coding. Moreover it is necessary to work with twice the frequency of the case without coding as two data are transmitted for only a useful one. Forcing the bus to 0 between each transmitted data introduces undesired transitions; therefore, the activity and thus the power consumption are increased.

In [17] authors propose another temporal coding called Code 1 which aims at reducing the data activity of the previous Code 0 technique. A two bit sequence on a wire is coded into a four bit sequence as shown in Table IV. Worst case transition of Code 1 will be $1+2r$ transitions. Results show a 6.7% power consumption reduction on the bus for a 1mm wire. But coded data activity of Code 1 is still slightly higher than the original data activity.

In order to avoid the activity increased by the coding, authors propose a final evolution of Code 1 called Code 2. For the same sequence of two bits, Code 2 introduces the coding of two consecutive blocks of four bits as shown in Table IV.

Table IV. Correspondence between original blocks and coding schemes

	Code 1	Code 2	
Original block	Coded block	1st coded block	2nd coded block
00	0000	0001	0000
01	0001	0011	1000
10	0011	0111	1100
11	0111	1111	1110

During the consecutive transmission of two sequences of two bits, the first sequence of coding and then the second are transmitted alternatively. The coded data activity is then the same as the original one with a worst case transition of$1+2r$. The bus consumption reduction is higher than both preceding techniques (up to 18.7% power consumption reduction on the bus). Unfortunately, authors of these temporal codings do not propose any codec power consumption evaluation.

Table V. Impact of the techniques on key parameters. This Table summarises advantages and drawbacks of the techniques presented in this Section as well as results in terms of power/energy consumption, activity and timing when they are numerically quantified in papers

Technique	Power (P) /Energy (E) variation	Propagation Time variation	Activity variation	Comments
Wire Spacing	Up to 40% Bus E↓	?	?	No extra circuitry ↑ Bus area
Shielding Vdd/GND	?	?	?	No extra circuitry Power lines distribution ↑ Bus area

Shileding AND	Up to 50% Bus E↓	↓	↑	↑ Bus area
Dynamic Voltage Scaling	4.6 times P↓ on average	↑	?	Complex hardware ↑ latency ↑ Bus area
Signal Skewing	?	5 to 20% ↓	?	Works fine with repeated buses No implemantation proposed
Gray Code	Up to 77 % Bus P↓	?	33% ↓ on instruction @ bus 12% ↓ on data @ bus	↑ latency with bus size ↑ Bus area
T0 Code	No dynamic consumption if seq @	?	↓	Complex hardware, 1 extra line Not efficient on chip
Bus Invert	Up to 50% I/O peak P↓ Up to 25% I/O average P↓	?	↓	Complex hardware, 1 extra line ↑ latency
Partial BI	?	?	62.5% ↓	Same comments as bus invert
Code Book	?	?	25 to 50% ↓	Consumption ↓ for extremely long wires (>7.5cm) Complex hardware, extra lines
Code 0	Up to 21.5% Bus E↑	↓	↑	Must work with 2.Fclk
Code 1	Up to 10.7% Bus E↓	↓	↑	Must increase bandwith 31.8%
Code 2	Up to 18.7% Bus E↓	↓	-	No implemantation proposed

Table V summarises advantages and drawbacks of the techniques presented in this Section. In this Table, results in terms of power/energy consumption, activity and timing are numerically quantified when presented in papers. If only tendency is shown, it is represented by arrows, otherwise it is represented by a question mark. In the next Section, our tool is used to evaluate quickly the impact on timing and power consumption of some of the techniques presented on Table V.

4.4. Optimization techniques analysis by Interconnect Explorer

In order to evaluate quickly which techniques have the best, Interconnect Explorer has been used on some of the previously presented techniques for data bus. Results have been obtained using stimuli data files (picture, music, speech) for a 65*nm* technology, on two bus metal layers (metal layer 2 and 4) for a length of 1*mm*. Table VI shows estimation results on activity, worst case capacitance, bus area (excluding codec area), propagation time and energy consumption on the bus. Partial Bus Invert is here Bus Invert applied to the last three least significant bits.

Table VI. Technique impact evaluation on key parameters using our tool. Table VI shows estimation results on activity, worst case capacitance, bus area (excluding codec area), propagation time and energy consumption on the bus. Results have been obtained using stimuli data files (picture, music, speech) for a 65*nm* technology, on two bus metal layers (metal layer 2 and 4) for a length of 1*mm*.

Data bus techniques	Activity variation	Worst case C_L	Bus area variation	Propagation time variation		Energy bus variation	
				Metal 2	Metal 4	Metal 2	Metal 4
Shielding GND	↓ 50%	$C_s + 2C_c$	↑ 107.8%	↓ 42.3%	↓ 42.6%	↓ 5.2%	↓ 6.2%
Shielding Vdd/GND	↓ 50%	$C_s + 2C_c$	↑ 107.8%	↓ 41.5%	↓ 41.8%	↓ 5.2%	↓ 6.2%
Shielding AND	↓ 18.6%	$C_s + 2C_c$	↑ 107.8%	↓ 42.3%	↓ 42.6%	↓ 1.9%	↓ 2.6%
Duplication	0%	$C_s + 2C_c$	↑ 107.8%	↓ 42.9%	↓ 43.2%	↑ 30.7%	↑ 30.2%
Bus Invert	↓ 11%	$C_s + 4C_c$	↑ 13.5%	0%	0%	↓ 7.9%	↓ 7.8%
Partial Bus Invert	↓ 19.5%	$C_s + 4C_c$	↑ 13.5%	0%	0%	↓ 11.4%	↓ 11.4%
Code 0	↑ 40%	$C_s + 2C_c$	0%	↓ 42.3%	↓ 42.6%	↑ 76.2%	↑ 75.8%
Code 1	↓ 5.1%	$C_s + 2C_c$	0%	↓ 40.8%	↓ 41.0%	↑ 42.2%	↑ 40.5%
Code 2	↑ 1%	$C_s + 2C_c$	0%	↓ 40.8%	↓ 41.0%	↑ 7.6%	↑ 6.4%

As shown in Table VI, these techniques always imply a reduction of the propagation time. On the other hand, the best activity and energy bus consumption reduction is obtained for Partial Bus Invert (11.4%). Therefore, an optimization could be to split the bus and apply coding techniques where consumption is the highest. This key issue for future optimization is discussed in the next Section.

4.5. Where does power consumption come from?

Bus power consumption is essentially dynamic and mainly depends on the capacitance of the wire and on its data activity. The top of Fig. 8 presents the activity of each wire on a bus for image data. The power consumption of least significant bits is given compared to the overall power consumption. As an example, the four least significant bits consumes 84% of the total power for image data. Finally the percentage of time that each transition class appears is given at the end of Fig. 8.

It can be noticed that power consumption is primarily located on the least significant bits (more than 50% power consumption for the last three bits). So, optimization techniques, which try to reduce power consumption on all bits, will have very reduced effect on most significant bits since their activity is weak. It is important to note that other data flows such as pictures, music, talking follow the same behaviour.

Fig. 9 illustrates, for Bus Invert and Code 2, the energy bus ratio of coded and non-coded data as a function of the number of bits on which the techniques are applied. For our example, the optimal case is when the technique is applied on 9 bits.

Another interesting result is that, techniques which have for aim to remove the worst cases of Table I (i.e. 1+3*r* and/or 1+4*r*), remove only a negligible part of the total transitions. For example, 1+4*r* transition type appears only 1% of time. Eliminating this worst case transition will therefore have a negligible impact on the global power consumption. Therefore removing these transitions will decrease bus propagation time, but will not systematically decrease power consumption. Indeed, 1+3*r* and 1+4*r* transitions are replaced by other coded transitions that can consume more power (a rising transition can be transformed into a falling one for instance).

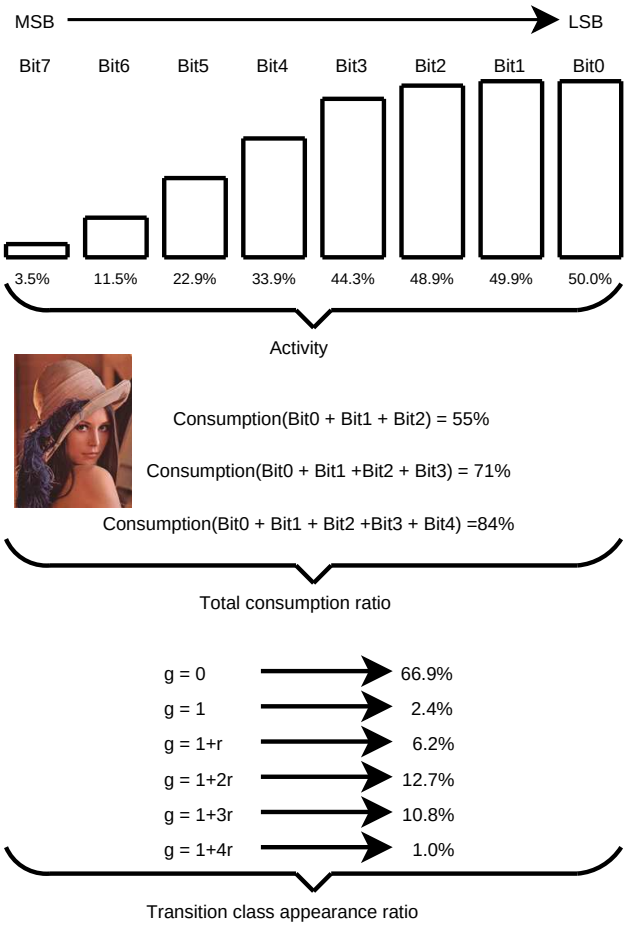


Fig. 8. Activity on each wire, consumptions of the sum of least significant bits compared to the overall consumption and transition class appearance for an image data flow

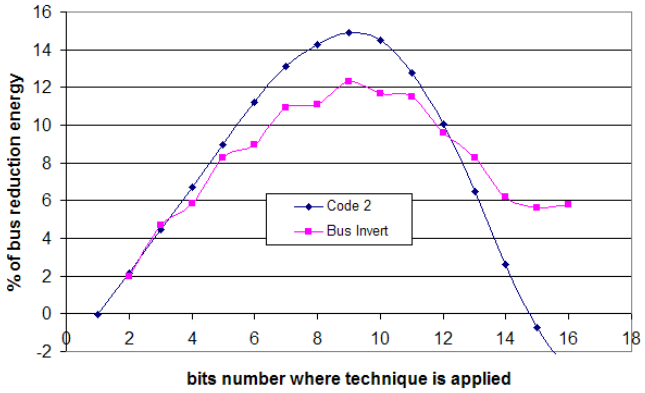


Fig. 9. Percentage of bus reduction energy in comparison with bit number where techniques are applied

Moreover, we noticed that, when results of power consumption optimization techniques are presented, authors do not always take into account the overhead power consumption introduced by codec. Most of the proposed techniques have a considerable hardware and thus power consumption overhead to carry out the data coding. To be efficient, it is essential that the power consumption overhead, due to the codec, remains lower than the gain generated by these optimization methods. Most of the methods have a codec complexity which is often important (register file, adders, multiplexers and so on). In [27], authors show that, in order to be efficient, techniques must be applied on buses with extremely long wires, which is in contradiction with wire length that can be found in system-on-chip.

5. CONCLUSION

This paper has first presented the physical parameters that are important for wires and bus modelling such as resistance and capacitance. Crosstalk effects have been discussed and their impacts on delay and power consumption shows that they must be taken into account to obtain accurate models at the bus level. Our delay and power consumption modelling methodology has been presented as well as our developed estimation tool (Interconnect Explorer). First results show that the classical transition pattern classification has to be carefully used in the case of power consumption.

A state of the art on delay and power optimization techniques has shown that techniques can optimize either time or power on the bus or both with a small gain for the best ones. Our tool has been used to compare techniques between them and also to underline the fact that techniques should be applied where activity is the strongest on buses (i.e. on least significant bits), and that eliminating worst case transitions has a negligible impact on the global power consumption.

To conclude, our future works on performance (time and power consumption) optimization techniques will be focused on the four key following issues:

- Do not only focus on 1+3r and 1+4r transitions.
- Focus on the lines where data activity is the most important of the bus (i.e. LSB).
- Try to avoid falling transitions as much as possible.
- Try to have a codec power overhead as weak as possible and therefore focus on very simple techniques.

Acknowledgements: This work has been supported by the European Union and the Brittany region in the context of “Programme Objectif 2 Bretagne 2000-2006”

References

1. N. Magen, A. Kolodny, U. Weiser and N. Shamir, Interconnect-power dissipation in a microprocessor. *Proceedings of the international workshop on System level interconnect prediction* (**2004***)*, pp. 7-13.

2. R. Ho, K. Mai and M. Horowitz, The Future of Wires. *Proceedings of the IEEE* (**2001**), Vol 89, pp. 490-504.

3. ITRS Technical report. International Technology Roadmap for Semiconductors, (**2006)**

4. W.J. Dally and J.W. Poulton, Digital Systems Engineering. Cambridge University Press (**1998**).

5. J.M. Rabaey, A. Chandrakasan, and B. Nikolic. Digital Integrated Circuits : A design perspective. Pearson Education (**2003)**,

6. A. Devgan, "Efficient Coupled Noise Estimation for On-Chip Interconnects," *IEEE/ACM International Conference on Computer Aided Design-Digest of Technical Papers* (**1997**), pp. 147–153.

7. J.M. Rabaey, A. Chandrakasan, and B. Nikolic, Digital Integrated Circuits : A design perspective, Pearson Education, Chapter 9, pp. 445–490, 2003.

8. C. Duan and S. P. Khatri, "Exploiting Crosstalk to Speed up On-Chip Buses," *Proceedings of the conference on Design,automation and test in Europe* (**2004**), p. 20778.

9. J.M. Rabaey, A. Chandrakasan, and B. Nikolic, Digital Integrated Circuits : A design perspective. Pearson Education (**2003)**, Chapter 5, pp. 179–233.

10. H.B. Bakaglu and J.D. Meindl, Optimal interconnection circuits for VLSI. *IEEE Trans on Electron. Devices* (**1985**), Vol. 32, No. 5, pp. 903–909.

11. A. Nalamalpu and W.P. Burleson, Optimal wire sizing and buffer insertion for low power and a generalized delay model. *Proceedings of the IEEE international conference on ASIC/SOC* (**2001**).

12. G. Chen and E.G. Friedman, Low-power repeaters driving RC and RLC interconnects with delay and bandwidth constraints*. IEEE Trans. on VLSI* (**2006**), Vol. 14, No.2, pp. 161–172.

13. W.C. Elmore, The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers. *Journal of Applied Physics* (**1948**), Vol. 19, pp. 55– 63.

14. S.P. Khatri, R.K. Brayton, and A.L. Sangiovanni-Vincentelli, Crosstalk Noise Immune VLSI Design Regular Layout Fabrics. Kluwer Academic Publishers (**1978**).

15. C.N. Taylor, S. Dey, and Y. Zhao, Modeling and minimization of interconnect energy dissipation in nanometer technologies. *Proceedings of the 38th conference on Design automation* (**2001**), pp. 754–757.

16. K. Hirose and H. Yasuura, A bus delay reduction technique considering crosstalk. *Proceedings of the conference on Design, automation and test in Europe* (**2000**), pp. 441–445.

17. J.M. Philippe, S. Pillement, and O. Sentieys, “Area efficient temporal coding schemes reducing crosstalk effects”, *Proceedings of the International Symposium on Quality Electronic Design* (**2006**), pp. 334–339.

18. L. Macchiarulo, E. Macii, and M. Poncino, Wire placement for crosstalk energy minimization in address buses. *Proceedings of the conference on Design, automation and test in Europe* (**2002**), pp. 158–162.

19. L. Shang, L. Peh, and N.K. Jha, Dynamic voltage scaling with links for power optimization of interconnection networks. *Proceedings of the 9th International Symposium on High-Performance Computer Architecture* (**2003**), pp. 91–102.

20. C.L. Su, C.Y. Tsu, and A.M. Despain, Saving power in the control path of embedded processors. *IEEE Design & Test of Computers* (**1994**), Vol. 11, pp. 24–31.

21. C.L. Su and A.M. Despain, Cache design trade-offs for power and performance optimization : a case study. *Proceedings of the international symposium on Low power design* (**1995**), pp. 63–68.

22. L. Benini, G.D. Micheli, E. Macii, D. Sciuto, and C. Silvano, Asymptotic zerotransition activity encoding for address busses in low-power microprocessor based systems. *Proceedings of the 7th Great Lakes Symposium on VLSI* (**1997**), pp. 77–82.

23. W. Fornaciari, M. Polentarutti, D. Sciuto, and C. Silvano, Power optimization of system-level address buses based on software profiling. *Proceedings of the 8th international workshop on Hardware/software codesign* (**2000**), pp. 29–33.

24. M.R. Stan and W.P. Burleson, Bus-invert coding for low-power I/O. *IEEE Trans. on Very Large Scale Integration Systems* (**1995**), Vol. 3, pp. 49–58.

25. Y. Shin, S.-IK Chae, and K. Choi, Partial bus-invert coding for power optimization of system level bus. *Proceedings of the international symposium on Low power electronics and design* (**1998**), pp. 127–129.

26. S. Komatsu, M. Ikeda, and K. Asada, Low power chip interface based on bus data encoding with adaptive code-book method. *Proceedings of the 9th IEEE Great Lakes Symposium on VLSI* (**1999**), pp. 368–371.

27. C. Kretzschmar, A.K. Nieuwland, and D. Muller, Why transition coding for power minimization of on-chip buses does not work. *Proceedings of the conference on Design, automation and test in Europe* (**2004**), pp. 10512–10517.

ANNEXE D

A NEW REAL-TIME METHOD FOR LEEWAY ESTIMATION BASED ON SAILBOAT DISPLACEMENT MODELING

Ronan Douguet
University of South-Brittany, UMR6285, Lab-STICC and Groupama sailing team, Lorient, France

Jean-Philippe Diguet
CNRS, UMR6285, Lab-STICC, Lorient, France

Johann Laurent
University of South-Brittany, UMR6285, Lab-STICC, Lorient, France

Yann Riou
Groupama sailing team, Lorient, FRANCE

Abstract: This paper presents new methods for real time estimation of leeway and ocean current, which contribute to boat displacements. We propose two solutions that rely on several types of Kalman filters. The first one uses the empirical leeway definition and allows validating the sailboat displacements modeling. The solution works properly if the error of the formula of leeway remains limited. In fact, the results of this model show the need to add a sensor to measure the boat drift, a DVL. So, the second solution takes advantage of this additional sensor and we compare three methods to linearize boat displacements, which are based on a closed-loop model including cascaded filters. These methods are tested on real data collected with a maxi multihull and the results validate the use of a DVL sensor for leeway estimation but also show that it requires the implementation of a complex and specific step of signal processing. So, our study demonstrates the relevancy of the closed-loop approach and shows that a solution, based on UKF filters, provides a relevant method to cope with accuracy and stability in case of sensor data outage.

Keywords: experimental methods, instrumentation, model testing, motions, optimization, performance assessment, performance prediction.

NOMENCLATURE

AWA	*Apparent Wind Angle*
AWS	*Apparent Wind Speed*
TWA	*True Wind Angle*
TWD	*True Wind Direction*
TWS	*True Wind Speed*

DVL	*Doppler Velocity Log*
IMU	*Inertial Measurement Unit*
BSP	*Boat Speed Measured by speedometer*
BSP_Lee	*Boat speed Corrected with the leeway angle*
BSP_Dvl	*Boat Speed Measured by DVL sensor*
COG	*Course Over Ground*
SOG	*Speed Over Ground*
Lat	*Latitude*
Lon	*Longitude*
HDG	*Heading*
LEE	*Leeway Angle*
k	*Leeway Correlation Constant*
CurD	*Current Direction*
CurR	*Current Rate*
KF	*Kalman Filter*
EKF	*Extended Kalman Filter*
UKF	*Unscented Kalman Filter*
X_g	*Navigation coordinates along x axis*
Y_g	*Navigation coordinates along y axis*
Vx_g	*Ground velocity of sailboat along x axis*
Vy_g	*Ground velocity of sailboat along y axis*
Ax_g	*Ground acceleration of sailboat along x axis*
Ay_g	*Ground acceleration of sailboat along y axis*
Vx_s	*Surface velocity of sailboat along x axis*
Vy_s	*Surface velocity of sailboat along y axis*
Ax_s	*Surface acceleration of sailboat along x axis*
Ay_s	*Surface acceleration of sailboat along y axis*
Vx_a	*Apparent velocity of sailboat along x axis*
Vy_a	*Apparent velocity of sailboat along y axis*
Ax_a	*Apparent acceleration of sailboat along x axis*
Ay_a	*Apparent acceleration of sailboat along y axis*
Vx_{lee}	*Leeway velocity along x axis*
Vy_{lee}	*Leeway velocity along y axis*
Ax_{lee}	*Leeway acceleration along x axis*
Ay_{lee}	*Leeway acceleration along y axis*
Vx_{cur}	*Current velocity along x axis*
Vy_{cur}	*Current velocity along y axis*

INTRODUCTION

Ocean racing history has always promoted and supported innovations and improvements of yacht design, sails and materials with the aim to improve sailboat performance. The sailing teams use to measure potential improvements with more and more sophisticated digital systems. These systems allow the teams to acquire data from sensors, to analyze the data flow and finally to make relevant choices (e.g. optimize the boat speed).

Today large volumes of data are acquired in real time via navigation processors, some of them are directly analyzed with these embedded systems and the others are based on post-processing techniques. These systems allow plugging several sensors (GPS, compass, anemometer-vane),

calibrating these sensors, computing and logging navigation data. For instance, the Racing Bravo processor, which has been developed for the American's Cup (Pons, 2004), allows the user to connect up to 255 sensors, to analyze boat performance and to display or store the results.

Now in ocean racing the boats are very close in performance, so it becomes necessary to measure the boat performance with a high degree of accuracy, even for few percent improvements.

The most important parameter to analyze is the wind since this is the engine of sailing boats. So since the last decade, wind measurement is one of the main subjects to performance analysis. In 1981, Arvel Gentry explained the errors in the wind measurements (Gentry, 1981). Currently, there are still the same problems but we have more sensors at our disposal. Since the wind sensor is fixed on the masthead, it is subject to several errors; some of them can be corrected properly but for the others it is more difficult. It is necessary to analyze these effects and to correct them (for instance the upwash, the leeway, the mast twist and so on).

In this paper, we propose a new method to compute the boat's global drift and especially new methods to improve the leeway and the ocean current measurements.

In the first part, we explain the different disturbances and corrections of the wind measurement. In a second part, we discuss about the modeling of the boat motions in order to compute the leeway and the current. We present our model based on set of Kalman Filters and the associated results tested in simulation. In the last section, we compare three modeling approaches to compute leeway and current with an additional sensor. Finally, we conclude the paper and give some perspectives.

WIND MEASUREMENT

Sensors

Different solutions exist to measure the wind but the sensor is always composed of two parts: the first one to measure the wind speed (anemometer) and the second one to measure the wind angle (vane).

The anemometer can use rotating cups, propeller or ultrasonic technologies. The advantages offer by the ultrasonic solution are a better accuracy than the propeller or rotating cups technologies in light airs and a measure less disturbed by the boat motions as it does not have any moving parts. On the other hand this kind of sensor is heavier and there is still less experiences regarding its reliability compared to the standard anemometer, so few ultrasonic sensors are in practice embedded on competition sailing boats.

Figure 55. Wind Vane Anemometer (B&G)

To measure the wind direction, ultrasonic sensors can also be used with the same reservations noted above. The other technology uses a mechanical vane and when coupled with a rotating cup system (cf. Figure 1), we obtain a complete system to measure wind speed and angle.

Wind Modeling

When the boat moves, the wind sensor measures an apparent wind. However, the skipper would like to know the true wind in order to analyze the boat performances. Figure 2 presents the relation between the apparent and true wind and the boat speed. To compute the true wind, we need to know the measured wind (speed and angle) and the boat speed. To obtain the wind speed and angle we use the sensors presented previously and a speed sensor to measure the boat speed. Usually, the speed sensors (usually 2 are used), named speedometer, are fixed on the hull one to starboard and the other one to port side (sometimes, the speedometer can be fixed on bulb keel but it is uncommon) and it measures the boat speed through the water.

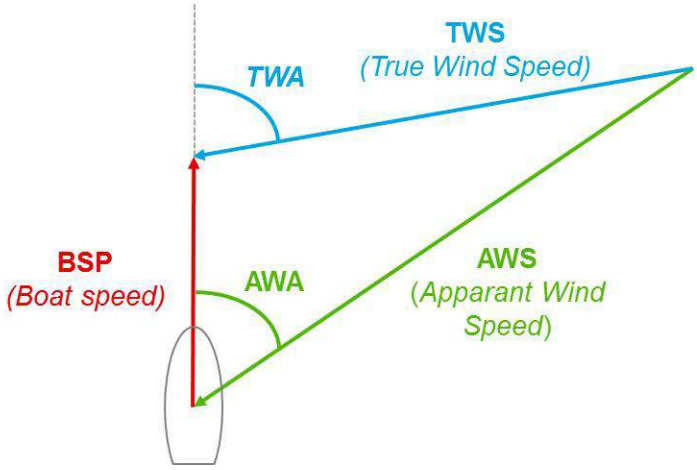


Figure 56. Wind Diagram

Unfortunately several parameters disrupt the true wind computation, since the wind measurements are affected by noise. We have to compensate for this noise in order to improve the true wind computation. The next section presents the different errors to be considered.

Wind measurements disturbances

The wind sensor, placed on the masthead, measures a disturbed apparent wind. The measurements are disrupted by several phenomena, such as the boat motions, the drift, the upwash effect and the wind shear, so it is necessary to correct the measurements. All disturbances and steps of the wind correction are explained in the next sections.

Offset Adjustments

There are several errors on the measured wind angle. First, the main wind sensors are not aligned properly with the boat axis. The measured wind angle is not the same on both tacks as shown in Figure 3. To correct this bias, we use simple computations based on parameters from the adjustment phase of the boat. When a rotating mast is used (e.g. wing mast), its rotation angle disturbs the measured wind angle. In this case, we need a mast rotation sensor to correct the wind done at runtime.

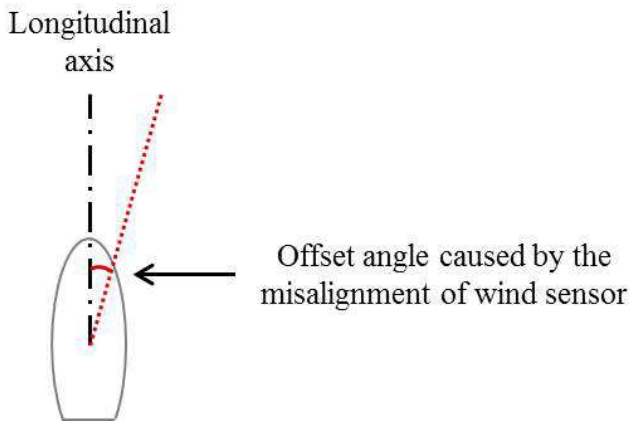


Figure 57. Misalignment of Wind Sensor

The last correction in this step is the twist of the mast. Indeed, while sailing the mast is subject to several forces that introduce an angle between the mast foot and the masthead as shown in Figure 4. A study performed by the Groupama sailing team on the Volvo Ocean Race boat shows that the twist angle is significant since it can reach up to 5 degrees.

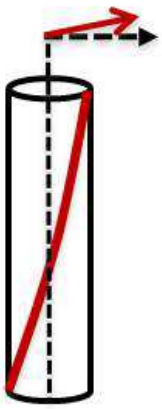


Figure 4. Mast Twist

Currently, any sensor can directly measure the twist angle. One of the solutions to compute this angle is to use two inertial measurement units (IMUs) (see Figure 5). Here, the aim is to compare the heading measurements on the masthead and on the mast foot; the difference between these two headings gives us the mast twist. The main problem of this solution is that the IMU is subject to important accelerations at the masthead and these measurements are therefore noisy.

Another solution to compute the twist angle is to use a video camera fixed on the masthead and a fixed mark on the deck. Then an image processing algorithm is used to determine the twist angle; the EPLF (Ecole Polytechnique Fédérale de Lausanne) uses this kind of technique and furthermore their system allows the user to study the sail shapes and foils (Bourgeon, 2010).

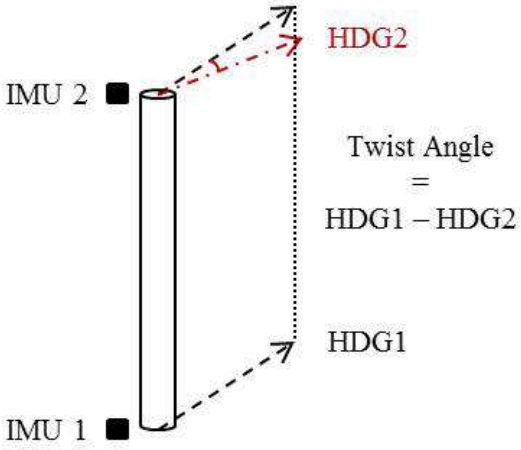


Figure 5. Twist Angle computation

The last solution is the use of strain gauges to measure the mast deformation. This solution is much lighter than the IMU or video camera system, but one of the two previous solutions is still needed during boat settings for calibration

Motion Correction

The boat motions of roll, pitch and trim create apparent wind components at the mast head that disrupt the wind measurement. Figure 6 presents the boat motions that must be corrected in order to improve the wind measurements; an IMU is used to filter these errors.

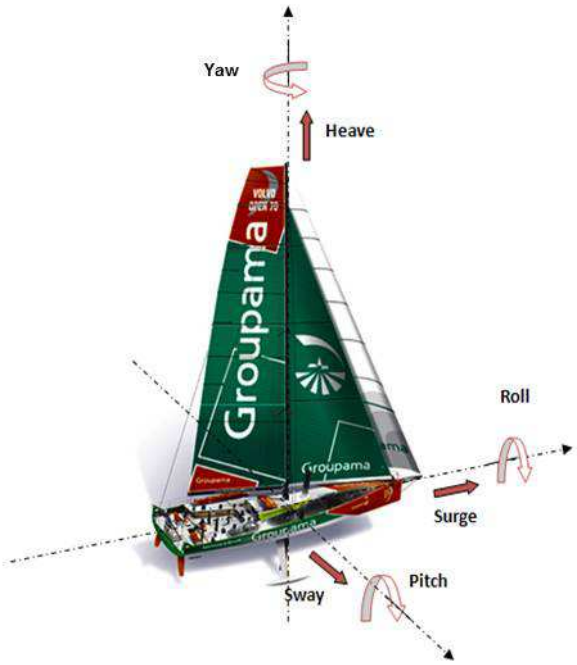


Figure 6. Boat Motions

From the angular velocities measured by IMU and from the mast height, we can compute the wind vector created by the boat motions at the mast head. This vector is computed in 2 dimensions because the wind sensor used measures the wind in 2 dimensions. So, to correct the wind measurements, we subtract this vector to the measured wind vector as shown in Figure 7. These corrections assume that the boat structure is rigid and therefore the measurements of IMU are correlated with wind measurements; the same assumption is used to correct wind measurements on catamarans (Rongere, 2010). The equations of this correction are available in appendix.

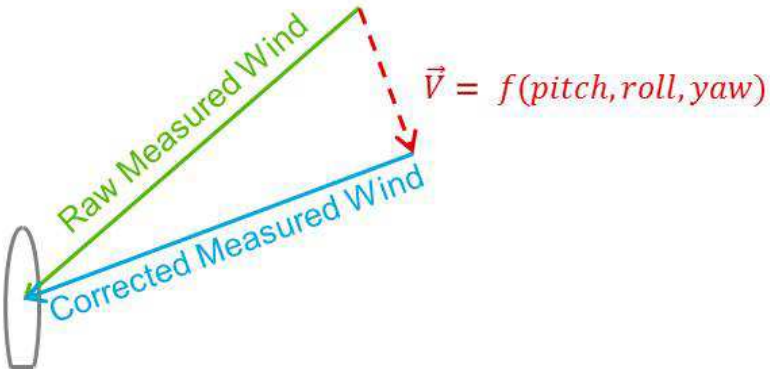


Figure 7. Correction of Boat Motions

The next step is to correct the attitude of the sailboat since the heel angle perturbs the measured wind. Currently, this correction is only applied on the wind angle since the influence of heel angle on the anemometer (rotating cups system) is not known (Brettle, 2001). The equations of this correction are available in appendix.

True Wind Computation

The vector computation of the true wind speed and angle is presented in Figure 8 so, we can see that we need to know the boat speed, the drift angle, the wind measurements and the heading. Here, the aim is to obtain the surface true wind since it is independent of the boat drift (due to leeway).

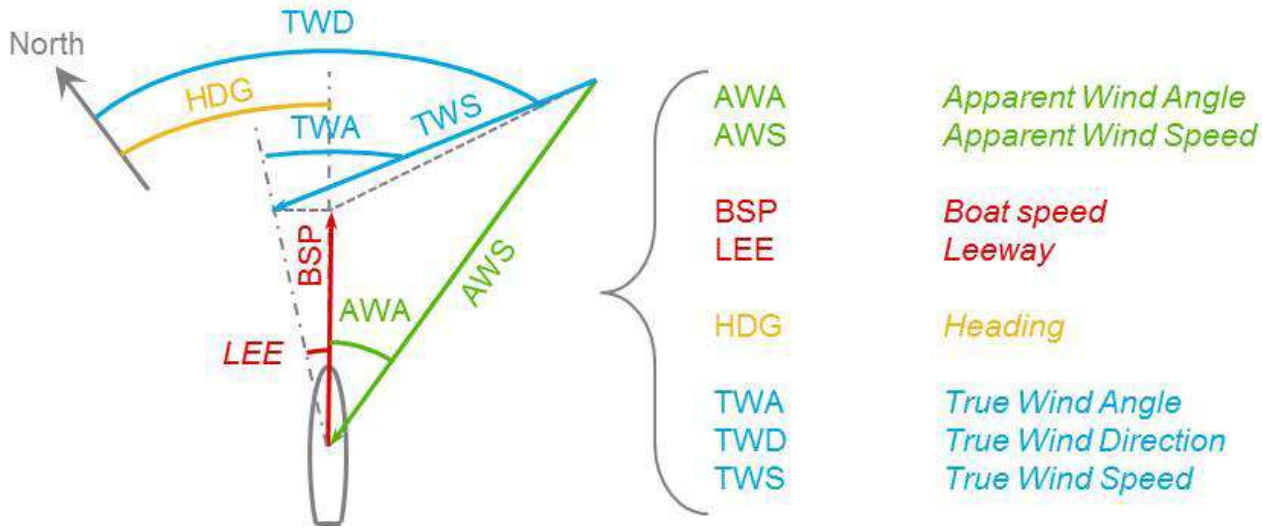


Figure 8. True Wind Diagram

In Figure 8, we can see that the leeway angle perturbs the true wind computation and the boat speed. Usually, the leeway angle is computed by an empirical formula defined by David Pedrick, 1981. This formula takes into account the heel angle, the boat speed and the leeway correlation constant.

$$\mathrm{LEE} = \frac{\mathrm{k} \times \mathrm{Heel}}{\mathrm{BSP}^2} \qquad (1)$$

Equation 1: Standard Leeway formula (Pedrick, 1981)

As the boat speed is also affected by the leeway a corrected value is taken into account and computed by the following formula:

$$\mathrm{BSP\_lee} = \frac{\mathrm{BSP}}{\cos(\mathrm{LEE})} \qquad (2)$$

Equation 2: Corrected Boat Speed

So with these corrections, a new true wind diagram can be defined; this diagram is shown in Figure 9. As now we have the true wind speed, we also compute the true wind direction from the heading and the true wind angle.

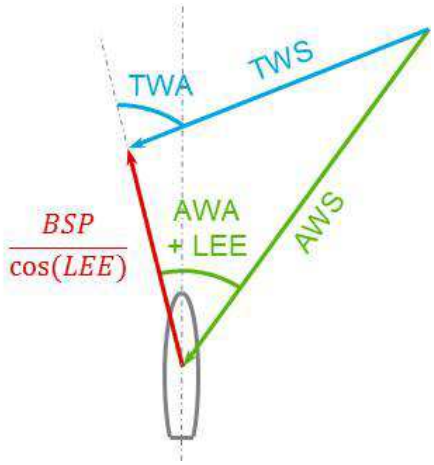


Figure 9. True Wind Simplified Diagram

Upwash Correction

The upwash effect represents the flow distortion at the mast head due to the sails. Indeed, vortex systems are created at the trailing edge of the sail as illustrated in Figure 10. So, when the sails are raised and generating lift, the measured wind at the masthead is further disrupted by the three-dimensional flow field around the sails (Gentry, 2006). To reduce this effect, the anemometer vane can be raised up to 1.5 meters above the masthead. Nevertheless a study realized, by the Groupama Sailing Team on VOR sailboat shown that the flow is still disrupted up to 16 meters above the masthead.

For correct the upwash effect, an offset value is defined for each true wind angle and speed computed. All these offset values are stored into several lookup tables generated from measurement campaigns since a table must be defined for each set of sails. The process to configure these tables is explained in a patent (Johnson, 2011).

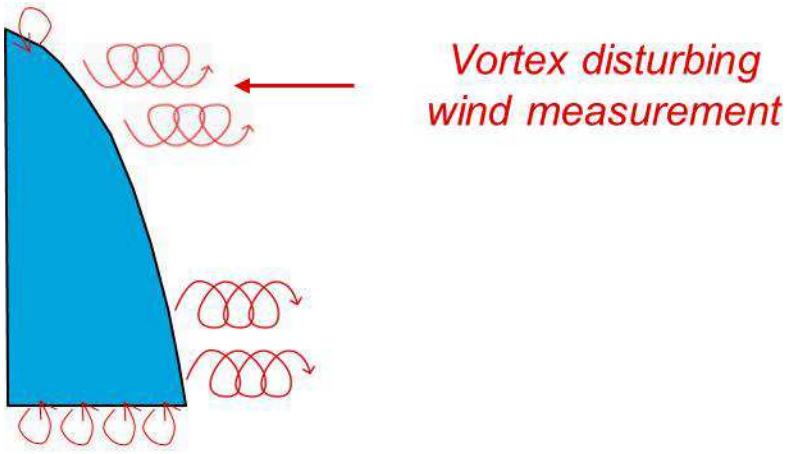


Figure 58. Upwash Effect

Wind Shear Correction

The wind shear, also called wind gradient, is a change in wind direction and speed depending on the altitude so in our case, it corresponds to a modification of the wind change between the masthead and the mast foot. This phenomenon is due to the friction in the earth's boundary layer near the sea

surface. The wind speed decreases when it gets close to the ground, as shown in Figure 11, thus in our case the breeze particles, on the water surface, will have the same speed as the water surface. It is hard to correct the wind shear because it depends on several variable parameters such as the sea and air temperatures and the wind speed and direction.

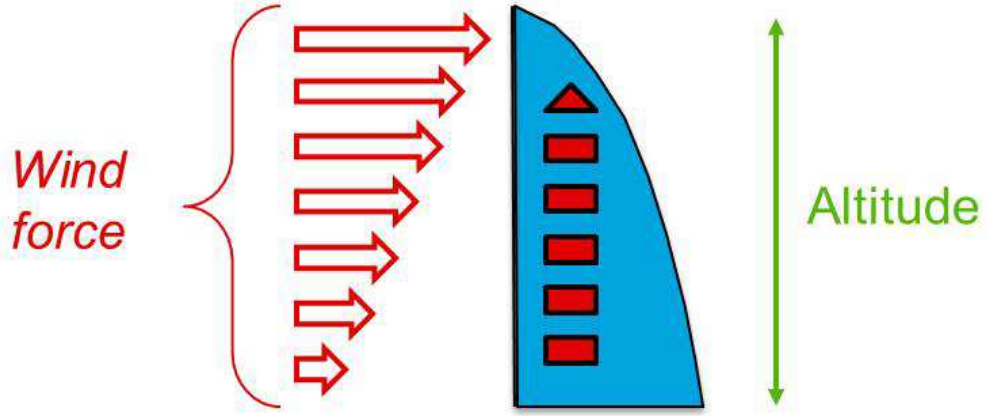


Figure 59. Decrease of Wind Speed Close to Water

Usually the wind shear is settled with an offset value applied to the wind angle and to the wind speed, but these parameters are empirically defined by the skipper. Today, there is not any fully reliable solution.

Navigation processor

Currently, to correct these disturbances we have just presented, we use a navigation processor. This system allows to plug and collect all data provided by sensors in order to compute the true wind. The figure 12 shows the wind corrected flow usually used in navigation processor.

In this paper, we have chosen to work on the leeway identification since this one is now computed empirically. In the next section, we explain the boat drift and the difficulty to measure the leeway.

Global Drift

On the one hand the wind propels the sailboat forward and on the other hand it makes the sailboat drifting sideways, this is the leeway. It is necessary to correct its impact on the measured wind, so the leeway angle must be estimated. Figure 13 shows that the sail force can be split up into two components: the propulsive force and the drift one.

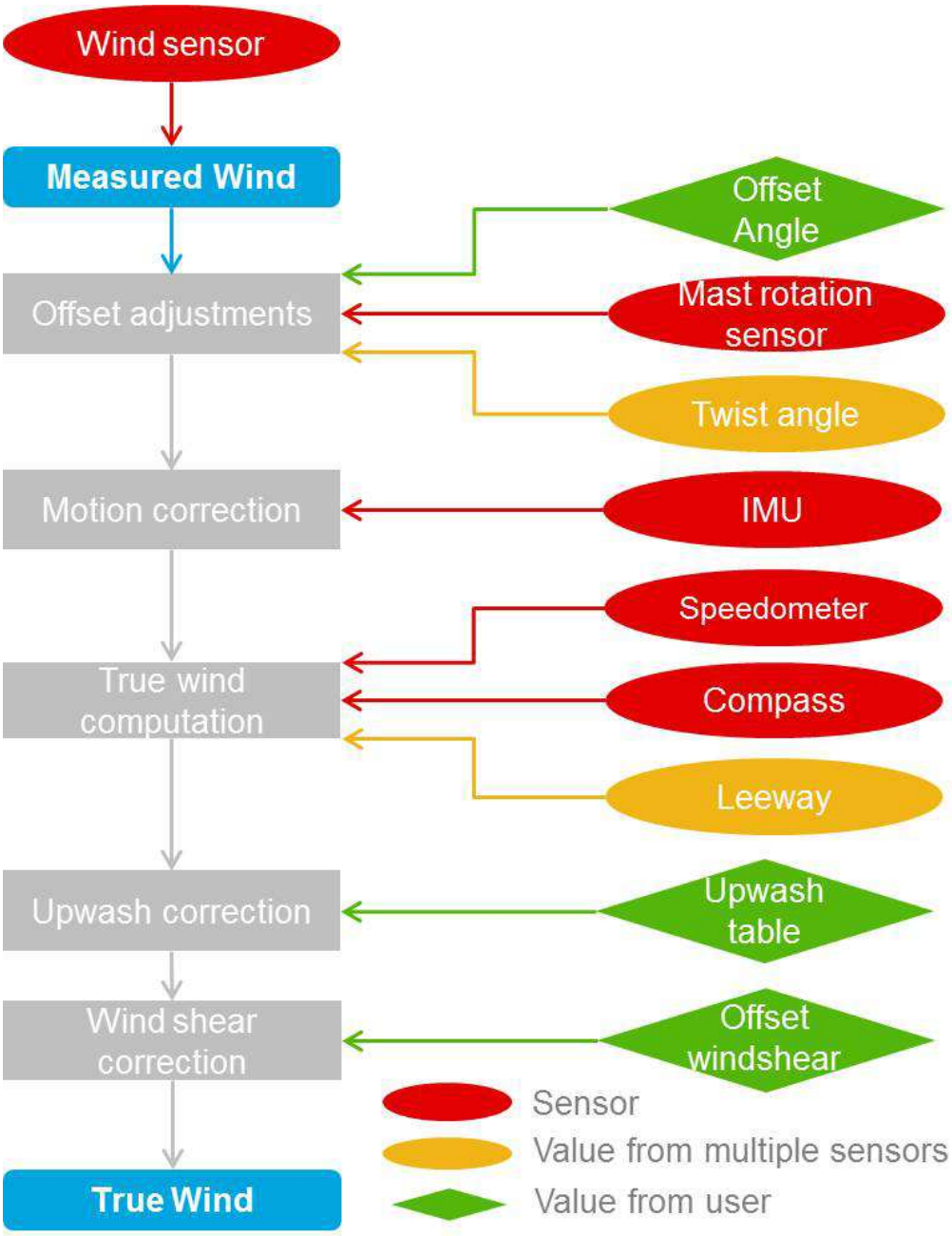


Figure 12. Wind corrected flow

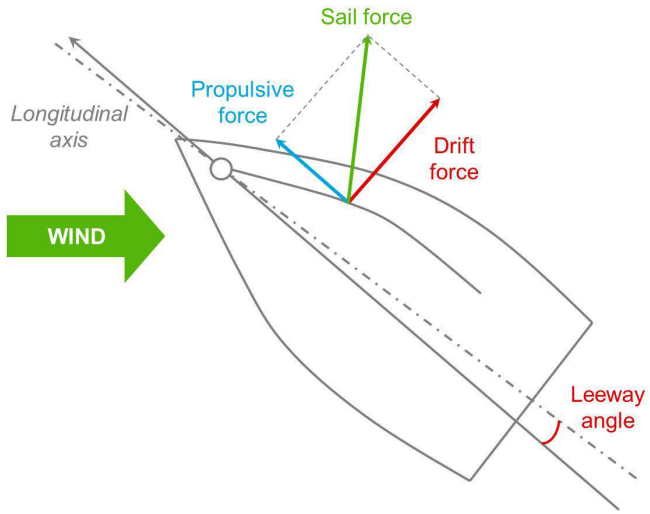


Figure 60. Decomposition of Sail Force

Currently, the leeway angle is determined by an empirical formula, which takes into account the boat speed, the heel angle and a constant. In addition, there are more parameters that affect the leeway such as the rudder angle, the daggerboards and foils height and so on. Moreover, this formula is defined for monohull boats so it is not applicable to multihulls. It is possible that several sailing teams have improved this formula but these results are not publicly available. Thus in this paper, we propose a new method to estimate the leeway in order to upgrade the measured wind.

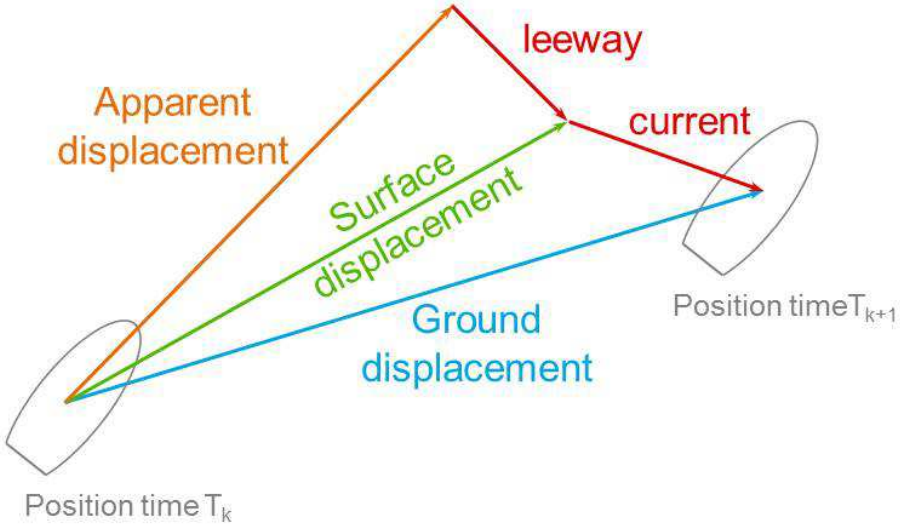


Figure 61. Decomposition of sailboat displacements

The main difficulty to identify the leeway is the boat also drifting with the ocean currents. The global drift is thus composed of the leeway and of the current as shown in Figure 14. So, there are three displacements: the apparent displacement, the surface displacement and the ground displacement.

The Figure 15 details the real displacements of a sailing boat. We can see that the apparent displacement defined by the heading (HDG) and the boat speed (BSP), the surface displacement

defined by the HDG plus the leeway (LEE) and the BSP_Lee and finally the over ground displacement defined by the Course Over Ground (COG) and the Speed Over Ground (SOG).

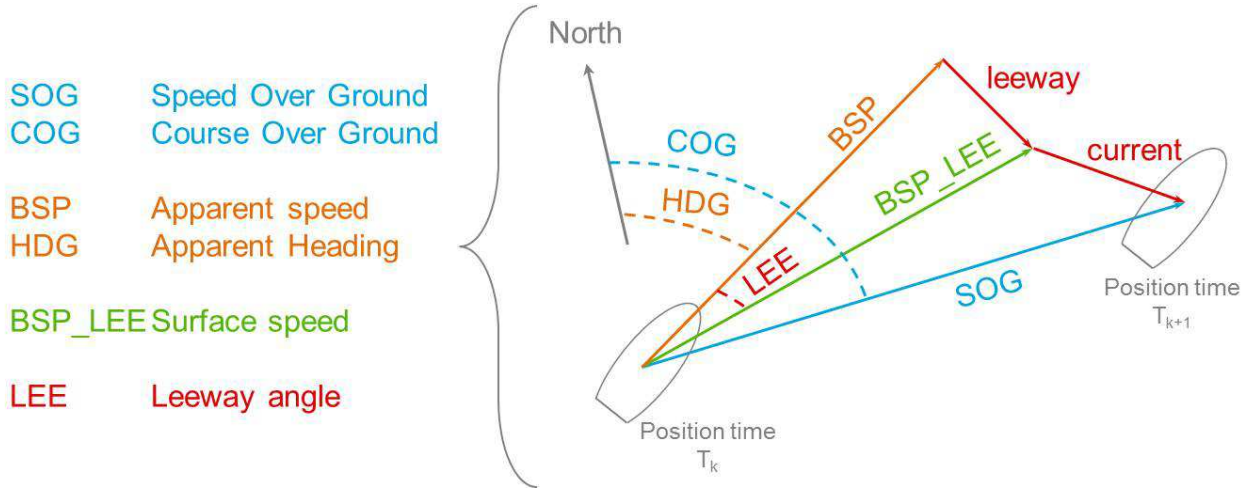


Figure 62. Analysis of sailboat displacements

Therefore it is difficult to analyze the global drift because we do not have current and leeway measures. However, it is possible to compute a current estimation from the main ocean current (Gulf Stream, Brazil Current…) or from the tide, but it is not accurate enough for competitive boat speed calculations. This estimation is not used to compute the leeway in real time but the skipper uses it to make optimal route choices.

By using some common sensors (GPS, Compass and speedometer) fixed on the boat, we can determine the global drift but we cannot dissociate the part of the drift due to the wind and the part due to the current. So in this work, we propose two different approaches to estimate the current and the leeway.

In the first case, we assume that the empirical formula of the leeway is noisy but coherent for a monohull boat; this model is tested on simulation dataset. The results achieved allows to validate the working of this model but shows that without knowing the uncertainty of leeway formula, the leeway does not converge to each time toward the reference value. Therefore, we propose to use an additional sensor to measure the leeway. Unfortunately, this new sensor is also disturbed by several phenomena. Thus, in the second case, we propose to use our model coupled with these new measurements in order to improve the leeway accuracy. In this part, different models are tested to determine the optimal solution.

LEEWAY ESTIMATION: FIRST APPROACH

Here, we present a solution to determine the leeway based on the empirical formula of leeway and from sensors measurements (GPS, speedometer and compass). First, we present the dataset simulator and then the model which is based on a set of Kalman filters. Finally, we conclude this part on the results and the interest of this model.

Simulation Dataset

To develop, test and validate this model, we have designed a simulation dataset. The aim is to have a credible dataset according to real measurements, so Gaussian noises, relevant to real sensor errors, are added to data. To set up the simulator, the user fixes the values of BSP, HDG, Heel, k, CurR, CurD for a predetermined period; the start position is also defined by the user. First, the simulator presented on Figure 16 computes the reference values: leeway angle (with the empirical formula), positions, speed and course GPS. Then, the white Gaussian noises are added to BSP, HDG, Heel and GPS data and from these noisy values the simulator computes the leeway, the speed and course over ground. The interest of developing a simulator is to provide noisy values and reference values; this is not the case with a real dataset (no reference values).

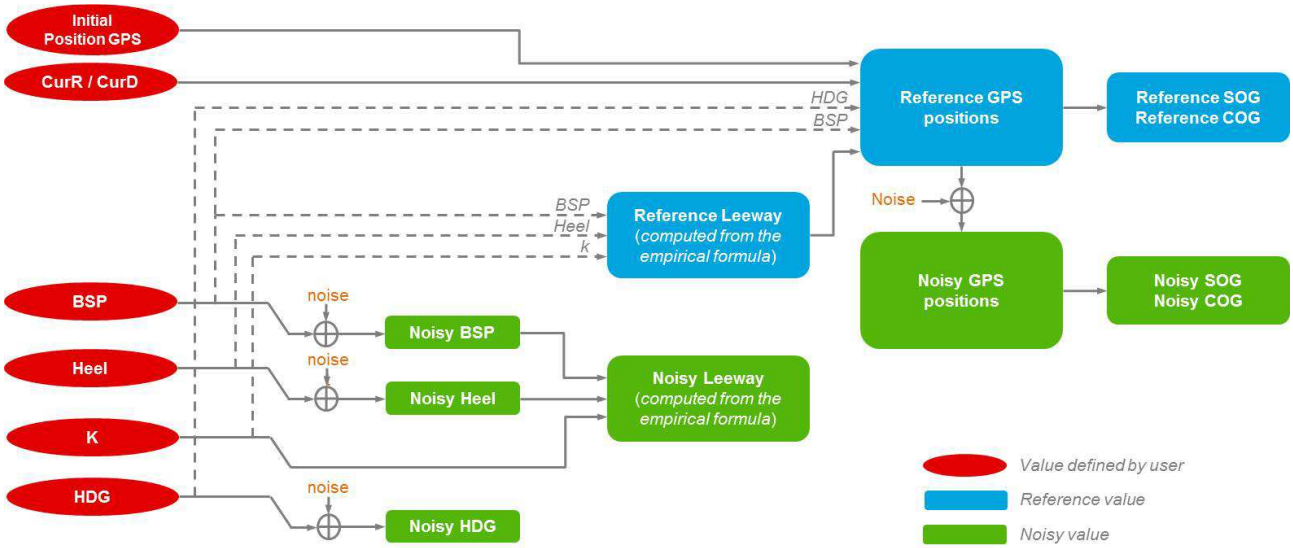


Figure 63. Data Simulator Diagram

Model

The global model of leeway and current estimations consists of two Kalman filters (KF). The Kalman filter is a recursive algorithm that allows to have an estimation of the system state vector using a modeling system and an observation vector (provided by experimental measurements). This allows estimating the useful information from one or more sensors whose measurements are disturbed by a white Gaussian noise. That's why, we have selected this filter; its operation is described in appendix B. On the other hand, the model uses two KFs in order to decrease the complexity of this one. Indeed, works (Bijker, 2008) show the utility of decomposing one filter in two filters of size lower to reduce the number of floating point operations. So, this model used in two phases (see Fig.17). The initialization phase computes a current through the empirical formula of leeway. Once the current is stabilized at the KF-1 output, it is used to provide an initial value of the leeway in the KF-2. Then a closed-loop model is implemented to jointly estimate the leeway and the current.

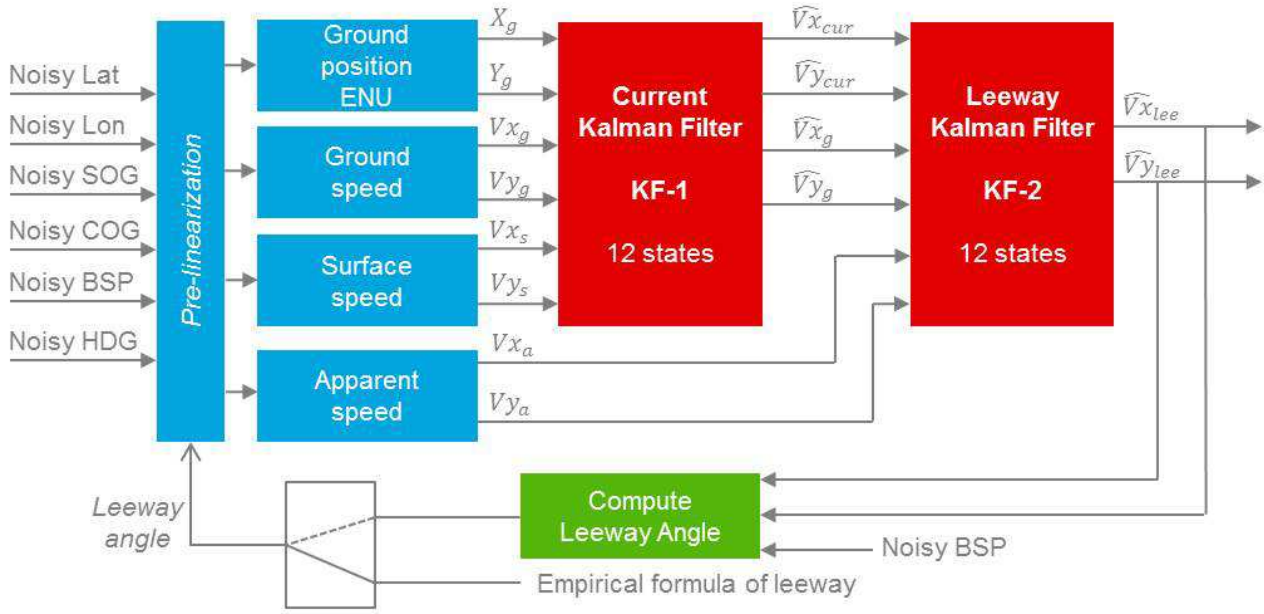


Figure 64. Model of First Approach

A step called "Pre-linearization" is required to use KF since this one is applicable only to linear problems; this step is explained in the next section.

Pre-linearization

To have a linear relationship in order to describe the sailboat displacements, we work with a Cartesian plane. The "Pre-linearization" allows to transform measurements in this plane. Thus, we convert GPS positions in a local tangent plane called ENU (East North Up) where the "Up" axis is unnecessary for the problem. Then, we compute the different vectors in this plane. These transformations are obtained by the following equations.

$$Vx_g = SOG \times \sin(COG) \qquad (3)$$

$$Vy_g = SOG \times \cos(COG)$$

Equation 3: Ground speed vector

$$Vx_s = \frac{BSP}{\cos(LEE)} \times \sin(HDG + LEE) \qquad (4)$$

$$Vy_s = \frac{BSP}{\cos(LEE)} \times \cos(HDG + LEE)$$

Equation 4: Surface speed vector

$$Vx_a = BSP \times \sin(HDG) \qquad (5)$$

$$Vy_a = BSP \times \cos(HDG)$$

Equation 5: Apparent speed vector

KF-1: Current KF

The KF-1 model allows to predict the ground displacement, the surface displacement and the current. The aim of this filter is to minimize the error between the ground and surface displacements and the current (cf. equation 6). Vx_{cur} and Vy_{cur}correspond to the ocean current vector in the ENU plane.

$$\in_x = Vx_g - (Vx_s + Vx_{cur}) \qquad (6)$$

$$\in_y = Vy_g - (Vy_s + Vy_{cur})$$

Equation 6: error of KF-1

The state vector is composed of ground displacement (position, speed and acceleration), surface displacement (speed and acceleration) and current displacement (speed) expressed in ENU plane. The observation vector contains the ground position, the ground speed and the surface speed; these vectors are illustrated on figure 18.

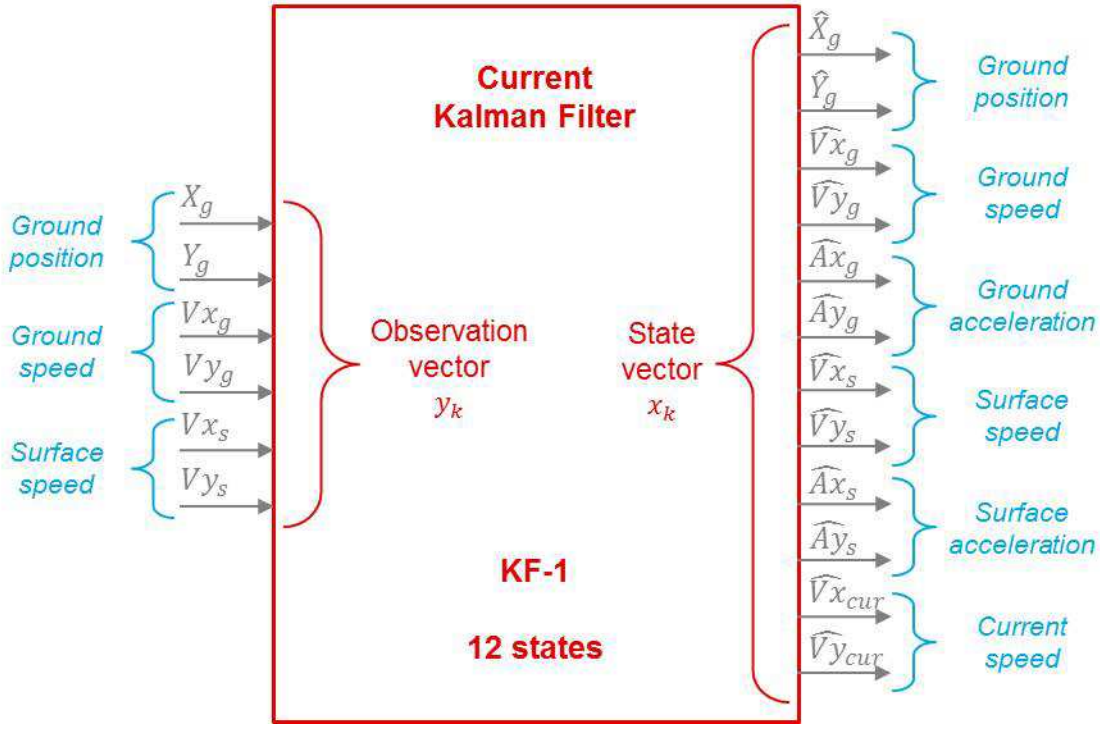


Figure 65. Current Kalman Filter KF-1

The prediction of the state is defined by the matrix F (cf. Equation 7). In the system, we consider the current, the surface and ground acceleration as constant over the sample period t_e. These assumptions must be taken into account to set the process noise covariance matrix Q. Thus, we measured the maximum variations on ground and surface accelerations and on current speed, respectively ($\in^2_{axg}$ and $\in^2_{ayg}$), ($\in^2_{axs}$ and $\in^2_{ays}$) and ($\in^2_{vxcur}$ and $\in^2_{vycur}$). From these measurements, we set the matrix Q (cf. Equation 8).

$$F = \begin{bmatrix} 1 & 0 & t_e & 0 & {t_e}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & t_e & 0 & {t_e}^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & t_e & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & t_e & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & t_e & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & t_e & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

Equation 7: State transition matrix (KF-1)

$$Q = \begin{bmatrix} (t_e^2.\in_{axg})^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & (t_e^2.\in_{ayg})^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & (t_e.\in_{axg})^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & (t_e.\in_{ayg})^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \in_{axg}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \in_{ayg}^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & (t_e.\in_{axs})^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & (t_e.\in_{ays})^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \in_{axs}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \in_{ays}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \in_{vxcur}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \in_{vycur}^2 \end{bmatrix}$$

Equation 8: Process noise covariance matrix (KF-1)

The relation between the state vector and the observation vector is also linear. It is expressed through the matrix H (see Equation 9). In this matrix, we establish the interactions between the current vector, the ground vector and the apparent vector (cf. Equation 10).

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \quad (9)$$

Equation 9: Observation Transition Matrix (KF-1)

$$Vx_{cur} = Vx_g - Vx_s \quad (10)$$

$$Vy_{cur} = Vy_g - Vy_s$$

Equation 10: Interaction between ground and surface vector

The last step is to set up the measurement noise covariance matrix R; this one has been determined by the measurement noises defined in the simulator. Indeed, a Gaussian noise was added on each measure used in the simulator. However, the "Pre-linearization" step is required to apply the Kalman filter; that's why to set up the matrix R, we need to calculate the measurement uncertainly propagated through the "Pre-linearization" step. Then, we obtain the matrix R (cf. Equation 11).

$$R = \begin{bmatrix} \sigma_{X_g}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{Y_g}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{Vx_g}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{Vy_g}^2 & 0 & 0 \\ 0 & 0 & 1 & 0 & \sigma_{Vx_s}^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & \sigma_{Vy_s}^2 \end{bmatrix} \quad (11)$$

Equation 11: Measurement Noise Covariance Matrix

At the filter outputs, we obtain the current vector in the ENU reference. It is used in input of the KF-2 filter to predict the leeway after the initialization step.

KF-2: Leeway KF

The KF-2 model allows to predict the ground displacement, the apparent displacement and the leeway. The aim of this filter is to minimize the error between the ground and apparent displacements, the leeway and the current (cf. equation 12). Vx_{lee} and Vy_{lee}correspond to the leeway vector in the ENU plane.

$$\in_x = Vx_g - (Vx_a + Vx_{cur} + Vx_{lee}) \quad (12)$$

$$\in_y = Vy_g - (Vy_a + Vy_{cur} + Vy_{lee})$$

Equation 12: error of KF-2

The state vector is composed of ground displacement (speed and acceleration), apparent displacement (speed and acceleration) and leeway displacement (speed and acceleration) expressed in ENU plane. The observation vector contains the ground speed, the apparent speed and the current speed; these vectors are illustrated on figure 19.

The prediction of the state is defined by the matrix F (cf. Equation 12). In the system, we consider the ground, the apparent and the leeway accelerations as constant over the sample period t_e. These assumptions must be taken into account to set the process noise covariance matrix Q. Thus, we measured the maximum variations on ground, apparent and leeway accelerations, respectively, ($\in_{axg}^2$ and $\in_{ayg}^2$), ($\in_{axa}^2$ and $\in_{aya}^2$) and ($\in_{axlee}^2$ and $\in_{aylee}^2$). From these measurements, we set the matrix Q (cf. Equation 8).

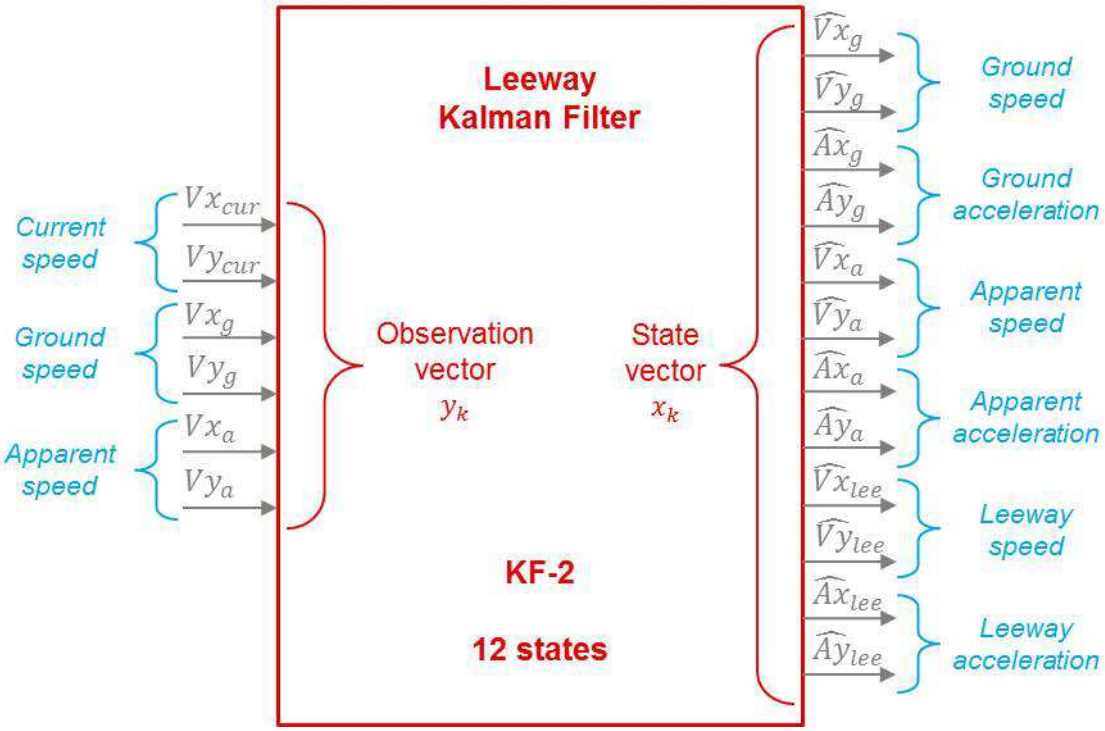


Figure 669. Leeway Kalman Filter KF-2

$$F = \begin{bmatrix} 1 & 0 & t_e & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & t_e & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & t_e & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & t_e & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & t_e & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & t_e \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (13)$$

Equation 13: State Transition Matrix (KF-2)

$$Q =$$

$$\begin{bmatrix} (t_e.\in_{axg})^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & (t_e.\in_{ayg})^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \in_{axg}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \in_{ayg}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & (t_e.\in_{axa})^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & (t_e.\in_{aya})^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \in_{axa}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \in_{aya}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (t_e.\in_{axlee})^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (t_e.\in_{aylee})^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \in_{axlee}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \in_{aylee}^2 \end{bmatrix}$$

Equation 14: Process noise covariance matrix (KF-2)

The relation between the state vector and the observation vector is also linear. It is expressed through the matrix H (see Equation 15).

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (15)$$

Equation 15: Observation Transition Matrix (KF-2)

The last step is to set up the measurement noise covariance matrix R; this one has been determined by the measurement noises defined in the simulator for the apparent displacement and by the statistics of KF-1 outputs for the ground and current displacements. To set up this matrix, we use the same method as KF-1 filter. So, we obtain the matrix R (cf. Equation 16).

$$R = \begin{bmatrix} \sigma_{vx_{cur}}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{vy_{cur}}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{Vx_g}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{Vy_g}^2 & 0 & 0 \\ 0 & 0 & 1 & 0 & \sigma_{Vx_a}^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & \sigma_{Vy_a}^2 \end{bmatrix} \quad (16)$$

Equation 16: Measurement Noise Covariance Matrix

At the filter outputs, we obtain the leeway vector in the ENU reference. After the initialization step, it is used to compute leeway angle which replaces the empirical formula of leeway in input of the KF-1 filter.

Results

To test this model, we configured the simulator with the parameters defined by the Table 1; these values were determined to be consistent with real values. The same parameters were used to set the matrices of measurement noise and the constant k of the empirical formula.

Table 3. Simulator configuration.

Parameter	Unit	White Noise	Value
GPS Pos.	meter	$N(0,2^2)$	⁄
BSP	knot	$N(0,2^2)$	5 to 20
HDG	degrees	$N(0,1^2)$	45
HEEL	degrees	$N(0,3^2)$	15
k	⁄	⁄	14.4
CurR	knot	⁄	1.3
CurD	degrees	⁄	170

As outlined earlier, this model works in two phases. The initialization phase computes a current by using the empirical formula of leeway and when this is stabilized we used it to compute the leeway and iterate the model.

Figure 20 shows the current rate and the current direction computed by the outputs of KF-1; in blue it is the reference and in red it is the estimation determined from the KF-1 filter.

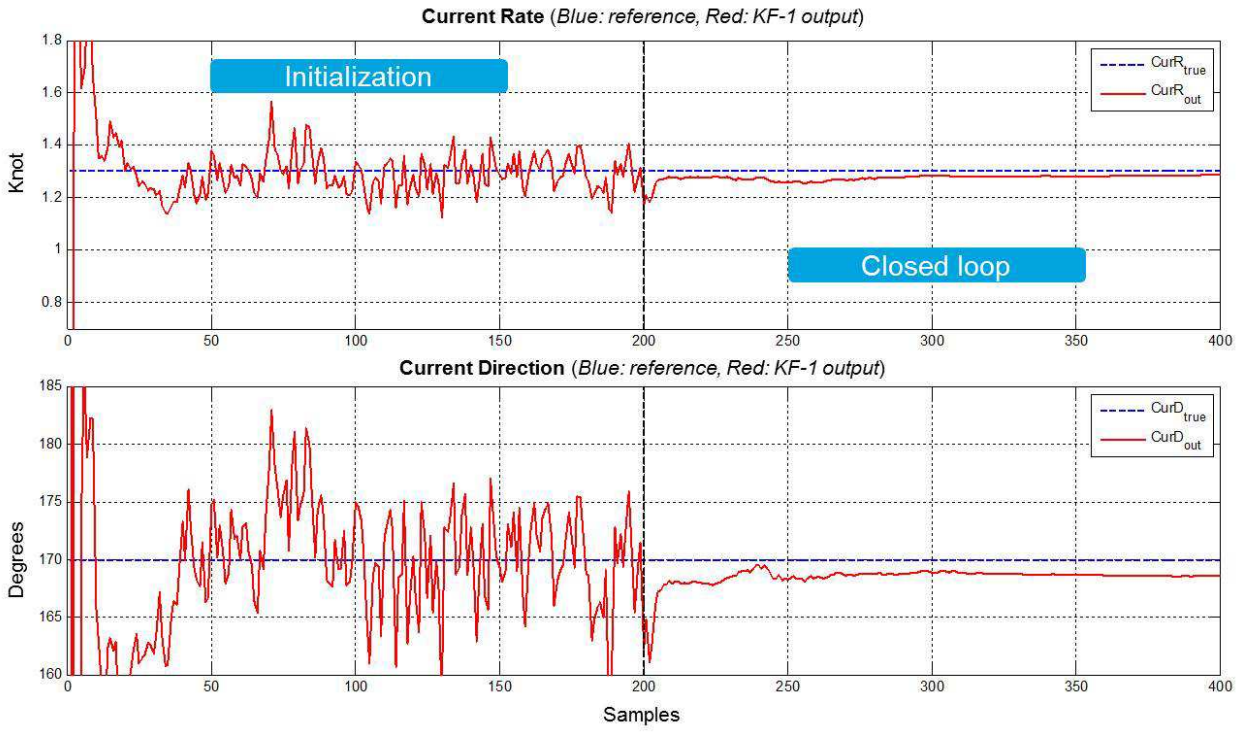


Figure 20. Results of ocean current estimation

Figure 21 shows the leeway estimation computed by the outputs of KF-2; the noisy leeway angle is in purple, the reference is in blue and the estimation of leeway determined from the KF-2 filter is in red.

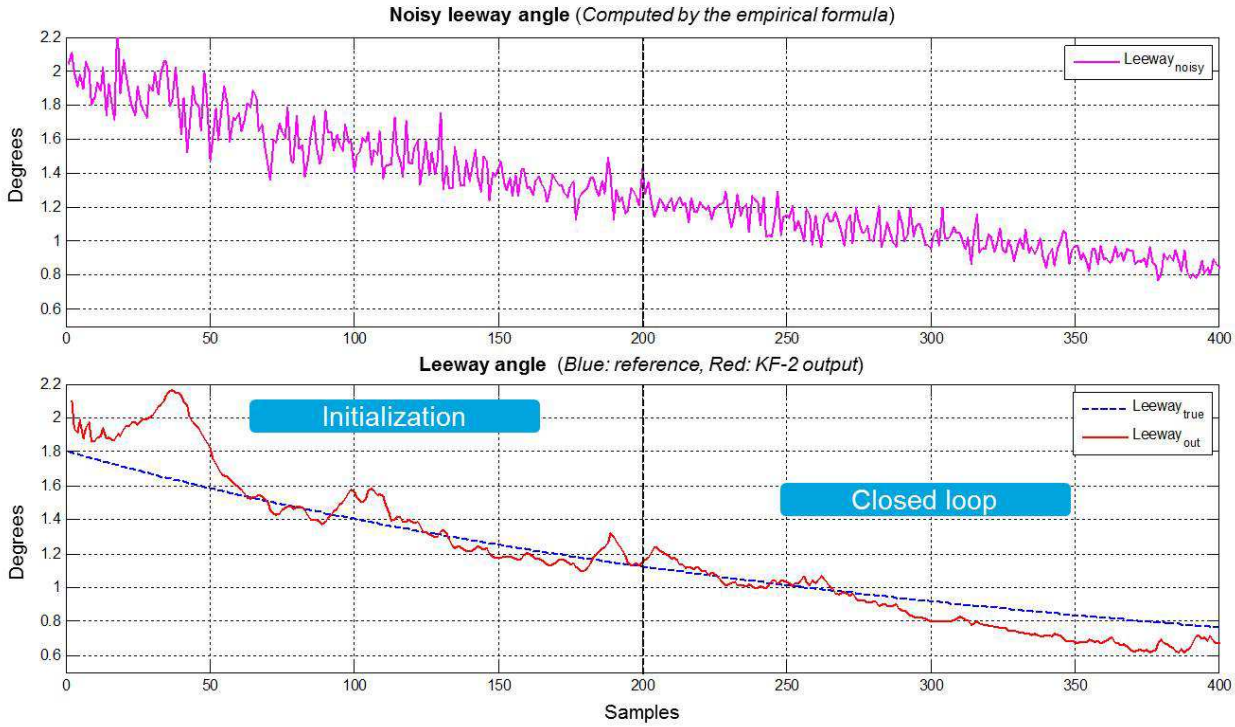


Figure 21. Results of leeway estimation

Analysis

The results of Figure 20 show that it is possible to converge towards the real values of current and leeway using the empirical formula during the initialization phase. However, for this model to converge properly, it is necessary that the error of the empirical formula remains limited. In reality it can be hard to obtain since the error on the k parameter is not known. Indeed, several tests have shown that with an error of fifteen percent on the k parameter the model does not converge.

Therefore, to improve the leeway calculation, we propose to use an additional sensor which measures the boat speed in several dimensions. In the next section, these new sensors are presented and several methods based on the boat displacements are proposed to improve the leeway and the current and to predict leeway and current estimation during sensor outages.

SECOND APPROACH WITH ADDITIONAL SENSOR

Sensor description

Several years ago, new type of sensors appeared which measure the speed in several dimensions: the Doppler Velocity Log (DVL). They are based on the acoustic technology and are able to measure the boat speed along its longitudinal and lateral axis. Recent sensor instances are smaller and lighter than previously and so can be mounted on sailboats. During the last Volvo Ocean Race, one sailing team had this type of sensor. To develop this DVL, a partnership had been established between the sailing team and the Nortek company. This DVL is composed of four beams which measure boat speed in four dimensions (cf. Figure 20).

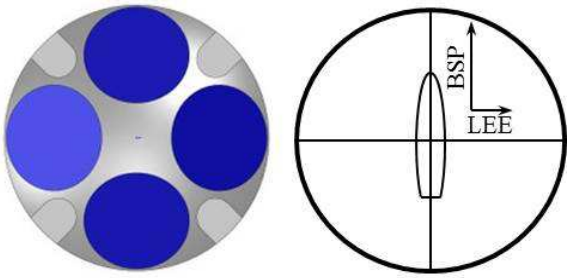


Figure 67. Acoustic Sensor Head

The main remaining problem of the DVL concerns its integration on the sailboat. When mounted on the hull, the sensor will be disturbed by the water flow close to the hull surface. To limit this problem the DVL can be fixed on the bulb of the keel. But, in this location, the sensor is also disturbed by the keel angle (canting keel), the keel twisting and the boat motions. For a multihull, the DVL can be mounted on the center hull or two sensors can be fixed on each hull (starboard and port). The disadvantage is that the DVL can be out the water for periods of time. In this case, there are DVL outages. In this study, we propose several models to predict leeway and current during DVL outages.

Figure 68. Explorer DVL (Teledyne RD Instruments)

A real dataset was provided by the "Team Sodebo Voile" containing DVL measurements from an Explorer DVL of Teledyne RD Instruments. This sensor is illustrated on Figure 21. On the Sodebo multihull, the DVL is mounted on the center hull.

Method description

The methods proposed here are based on the prediction of boat displacements. The aim is to upgrade the leeway value, to compute a current and to predict the leeway and current during DVL outages.

As the system is nonlinear, it was not possible to directly use a KF; that is why, in the first model (as in the first approach), a pre-linearization step is required. Two other models are tested in order to determine the optimal solution in terms of accuracy, stability and computational complexity. Thus, we proposed to use two additional filters: the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF). Indeed, in this case of nonlinear systems, the most common approach is to use EKF (cf. Appendix C), which linearizes all nonlinear models so that the traditional linear KF can be applied. Another solution is to use the UKF (cf. Appendix C) which is an improvement proposed by Julier and Uhlman in 1997. So, we compared three different models.

To compare these models, the same noise measurement was used to initialize these models. For the DVL, speedometer and compass, the error provided in the datasheet was considered. One

approximation was done on the COG and SOG: errors have been determined by computing COG and SOG between two GPS coordinates. Between these two coordinates, the noise measurement was considered to be independent.

The next sections describe the three proposed models and their results.

Model 1: KF with pre-linearization

This model (see Fig. 23) is similarly to the previous model; it is based on a pre-linearization step and on the two KFs previously described (KF-1 and KF-2). We have just replaced the empirical formula of leeway by the DVL measurements. Indeed, in this model, we used the surface speed (BSP_dvl) and leeway angle (LEE) measured by DVL and the apparent speed (BSP) measured by the speedometer.

Our system works in two phases: a first phase in which it estimates the current and leeway from DVL measurements and a second phase in which it is looped back on itself in order to continue to predict the current and leeway during DVL outages.

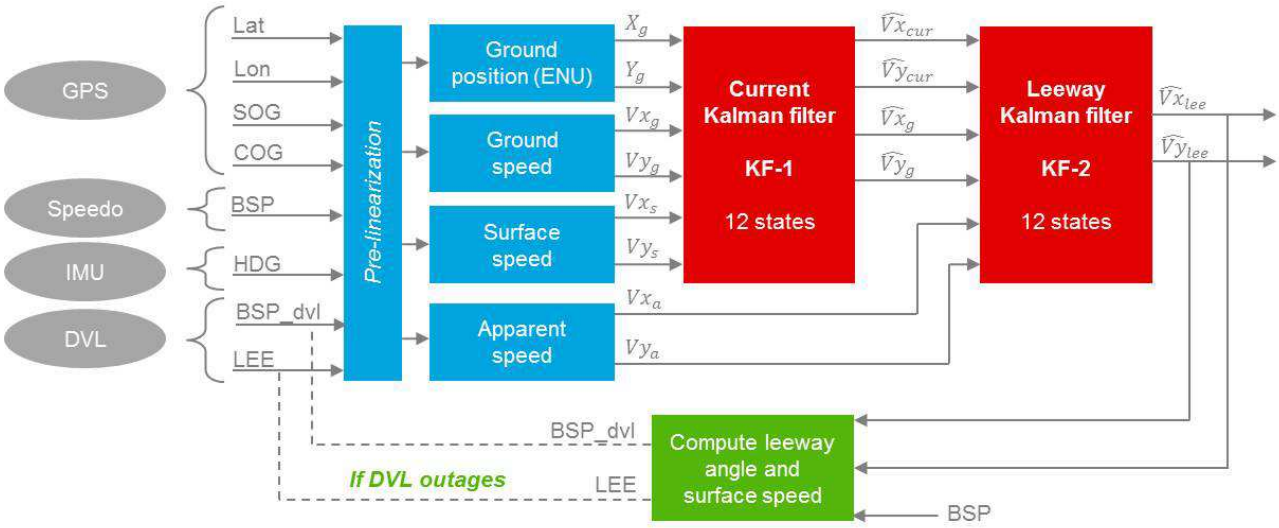


Figure 69. Diagram of Model 1

This model and the two other solutions have been tested in various data sets. In this study we present simulation results over a 400 second period that includes DVL outages with a 100 second period. Figure 23 presents the results of KF-1 (in blue: raw current computed by DVL and BSP measurements, in red: current computed by KF-1). Figure 24 shows the results of KF-2 (in blue: DVL measurements, in red: BSP_DVL and leeway computed by KF-2). The following models are tested and presented in the same way.

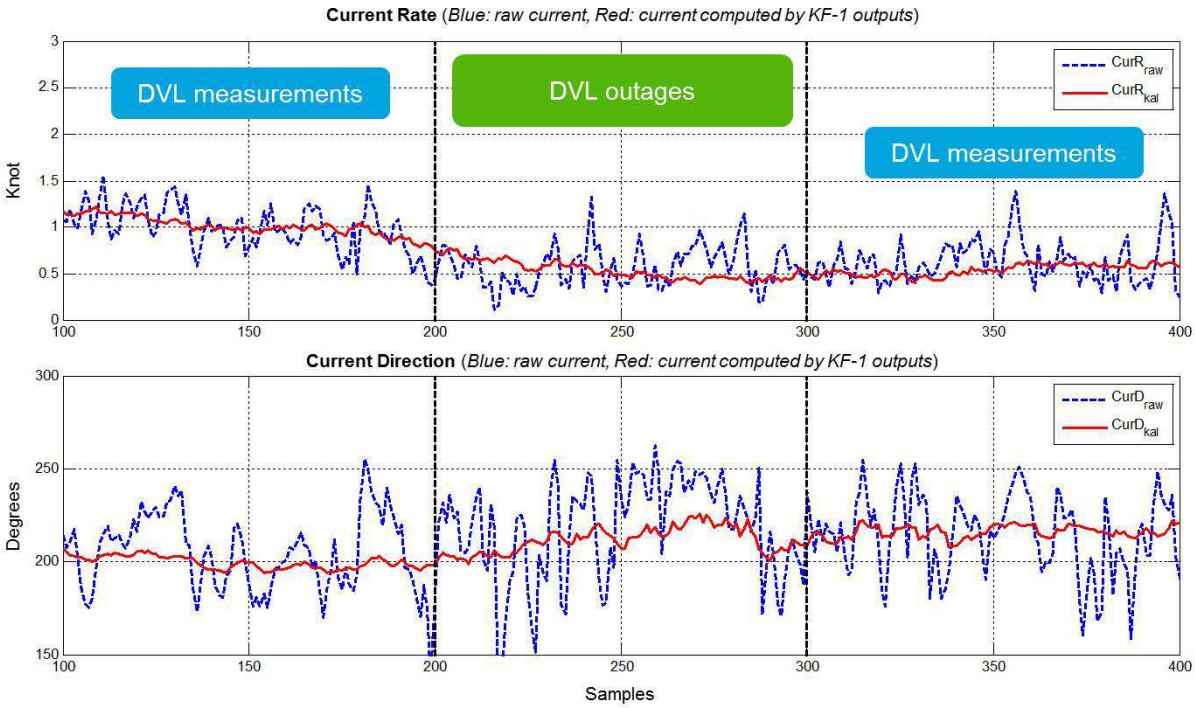


Figure 70. Current estimation of Model 1

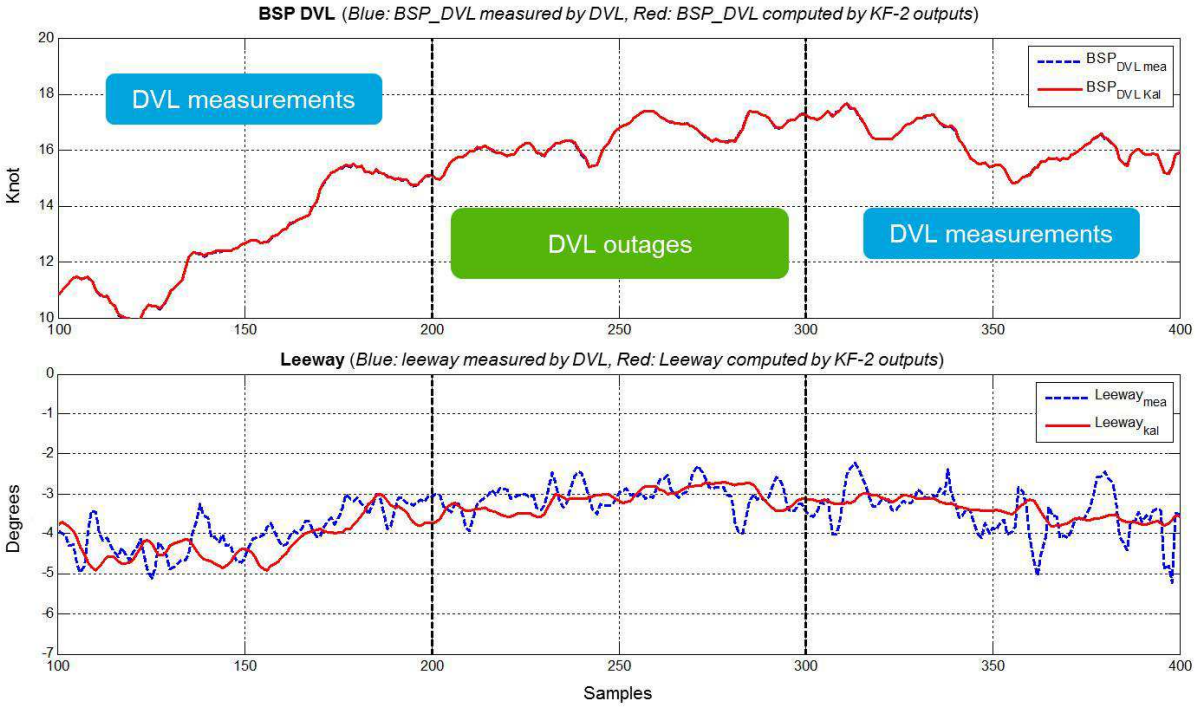


Figure 71. Leeway and BSP estimation of Model 1

Model 2: EKF

The model 2 is based on two EKF (cf. Figure 26); these two filters use directly the sensor measurements since these can linearize nonlinear models. As previously, the first filter EKF-1 allows to estimate the ocean current and EKF-2 allows to estimate the leeway.

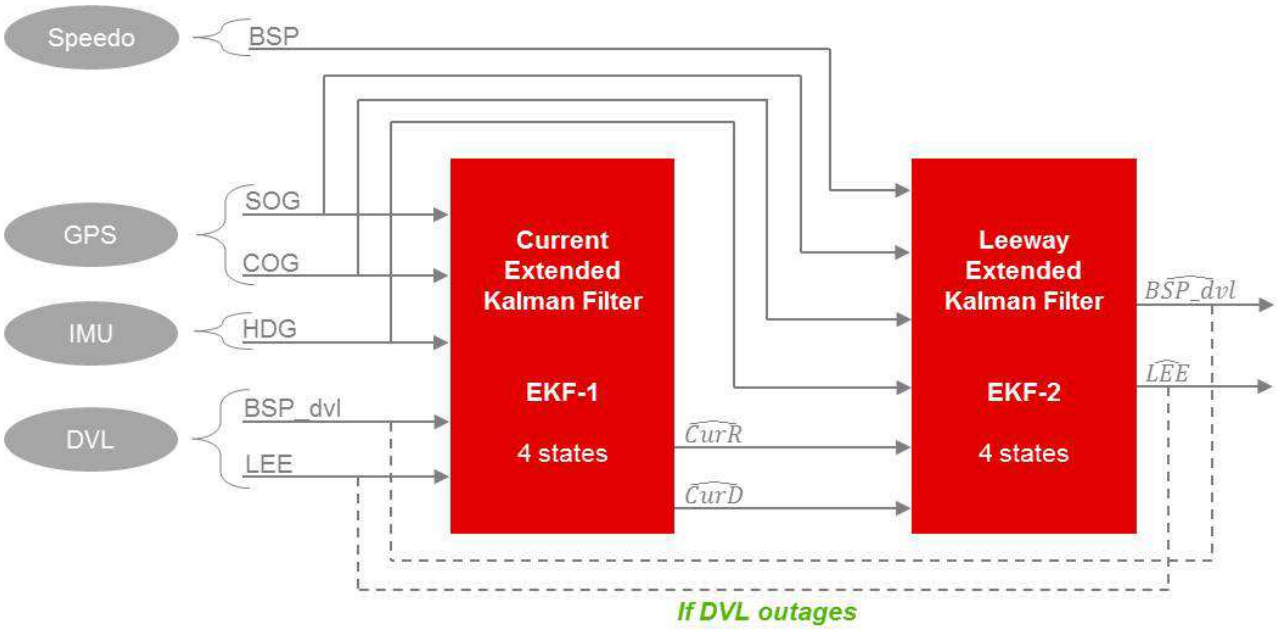


Figure 72. Diagram of Model 2

In these two EKFs, the interactions between the state vector and the measurement vector are defined by Equation 17. So, the aim of these filters is to minimize the error between the ground, surface and current speed (cf. Equation 18).

$$CurR.\cos(CurD) = SOG.\cos(COG) - BSP\_dvl.\cos(HDG + LEE) \quad (17)$$

$$CurR.\sin(CurD) = SOG.\sin(COG) - BSP\_dvl.\sin(HDG + LEE)$$

Equation 15: Interaction between measurements

$$\epsilon = \overrightarrow{V_{ground}} - \left(\overrightarrow{V_{surface}} + \overrightarrow{V_{current}}\right) \quad (18)$$

Equation 18: error of EKF-1 and EKF-2

On the other hand, the EKF-1 and EKF-2 reduce the dimension of the state vector at 4 since we use directly the sensor measurements but they add Jacobian computation (which allows to linearize the problem). The observation and state vectors on these two filters are described on Figure 26; all system equations are defined in Appendix D and E.

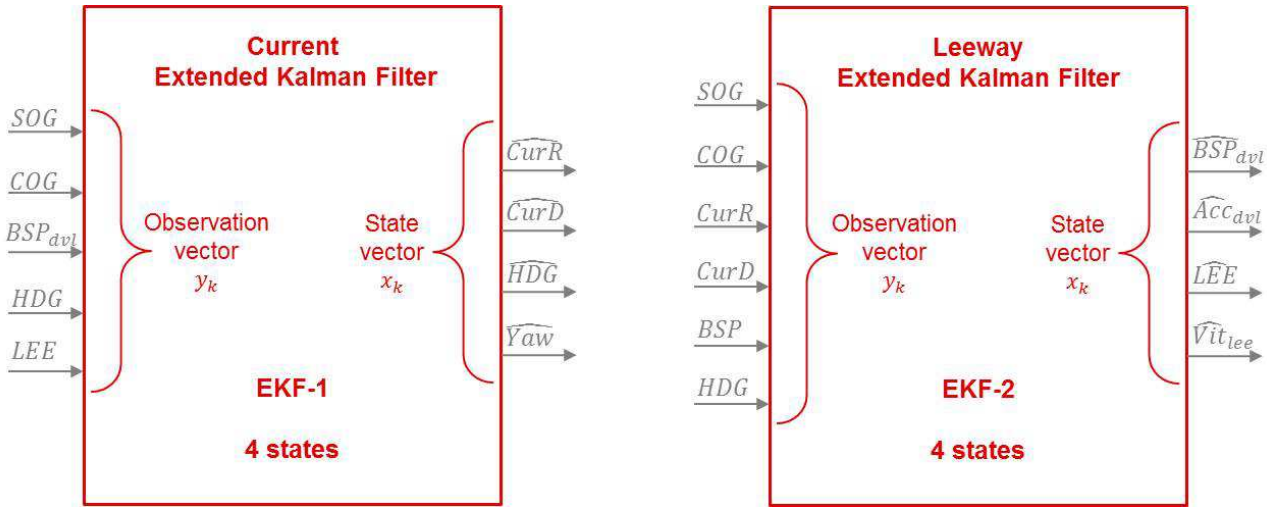


Figure 73. Observation and state vectors

As the previous model, the results show the system work during these two phases: with DVL measurements and with DVL outages. Figure 27 presents the results of EKF-1 (in blue: raw current computed by DVL and BSP measurements, in red: current computed by EKF-1). Figure 24 shows the results of EKF-2 (in blue: DVL measurements, in red: BSP_DVL and leeway computed by EKF-2).

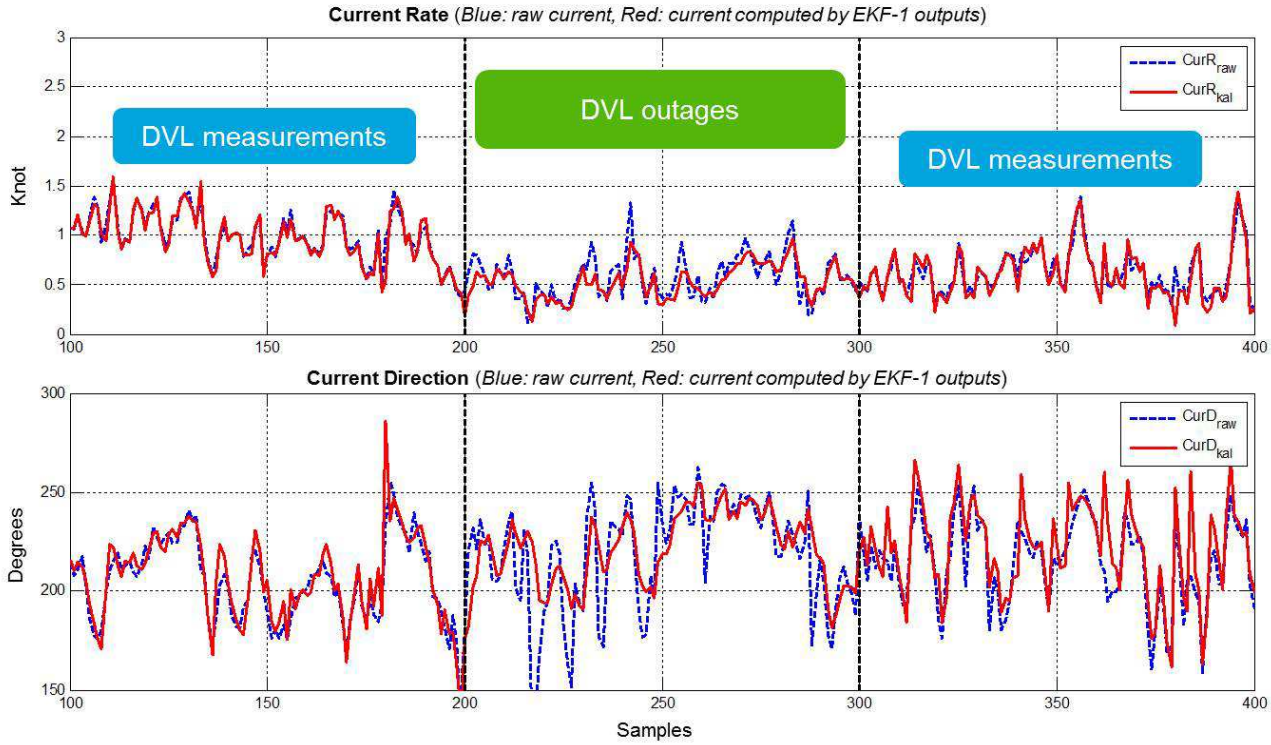


Figure 74. Current estimation of Model 2

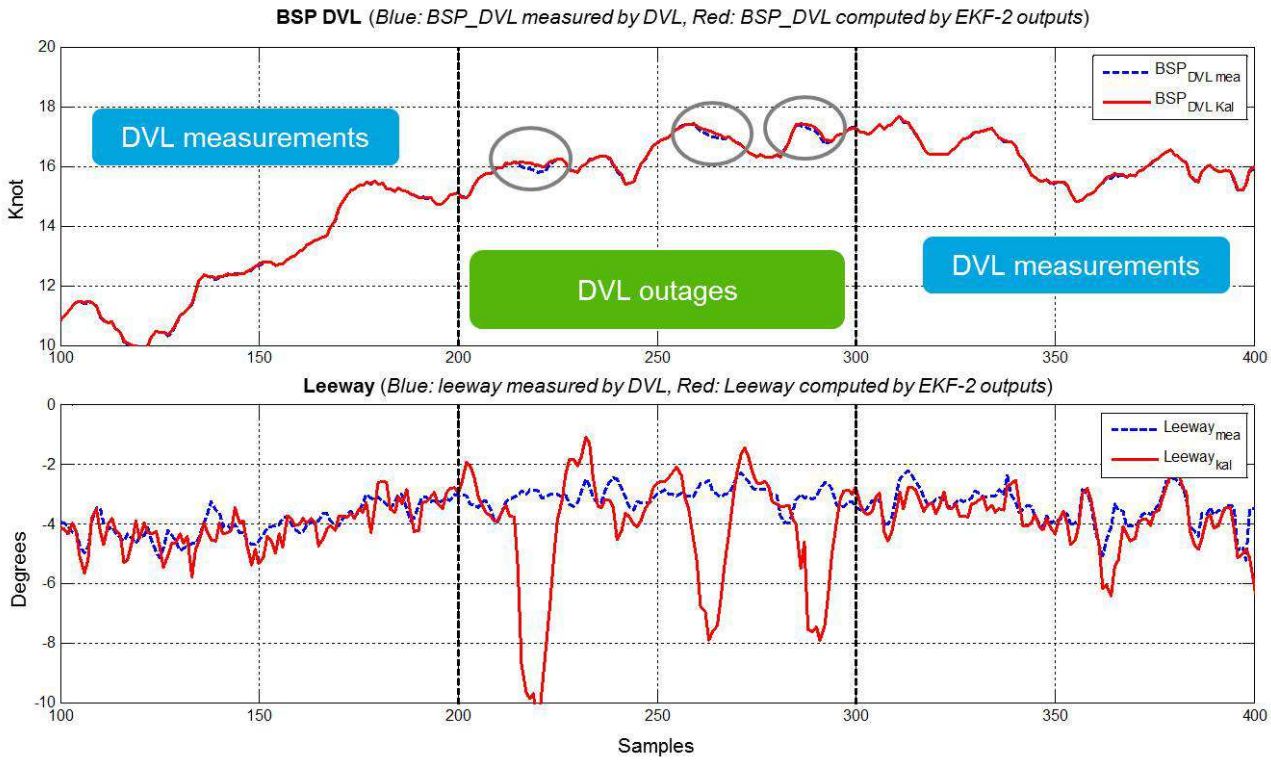


Figure 75. Leeway and BSP estimation of Model 2

Model 3: UKF

The model 3 is similar to the model 2. The EKF are just replaced by UKF (cf. Figure 29). All system equations are described in Appendix C and D.

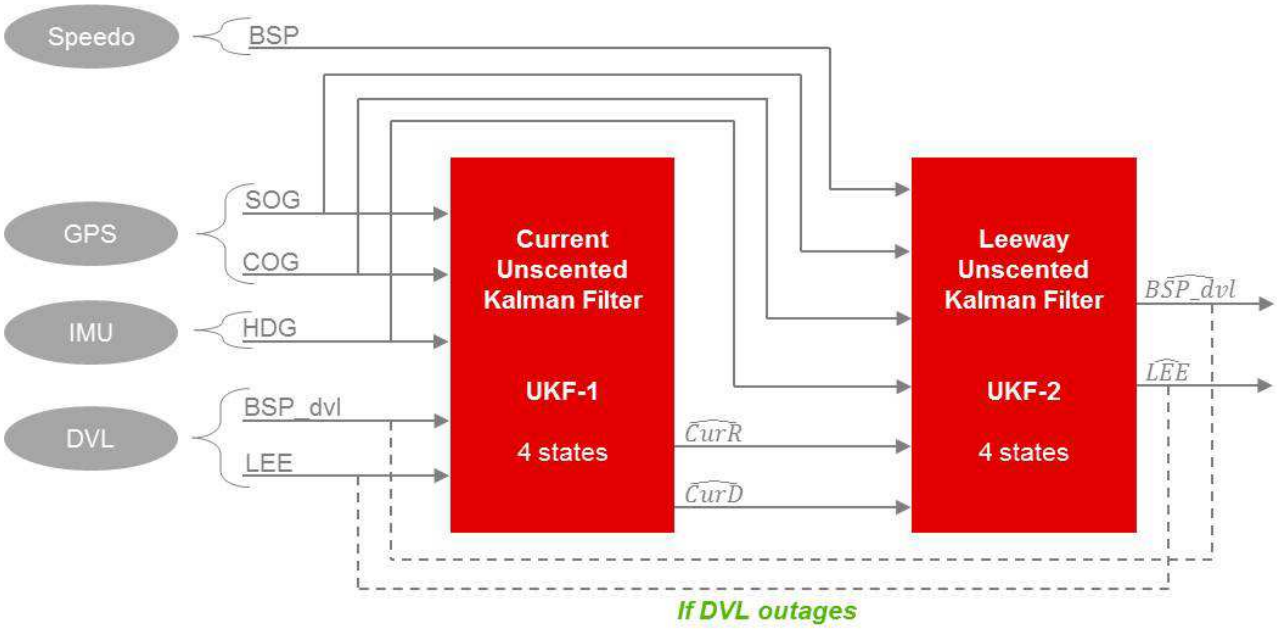


Figure 76. Diagram of Model 3

The results of UKF-1 are presented on Figure 30 (in blue: raw current computed by DVL and BSP measurements, in red: current computed by UKF-1) and the results of UKF-2 are presented on Figure 31 (in blue: DVL measurements, in red: BSP_DVL and leeway computed by UKF-2).

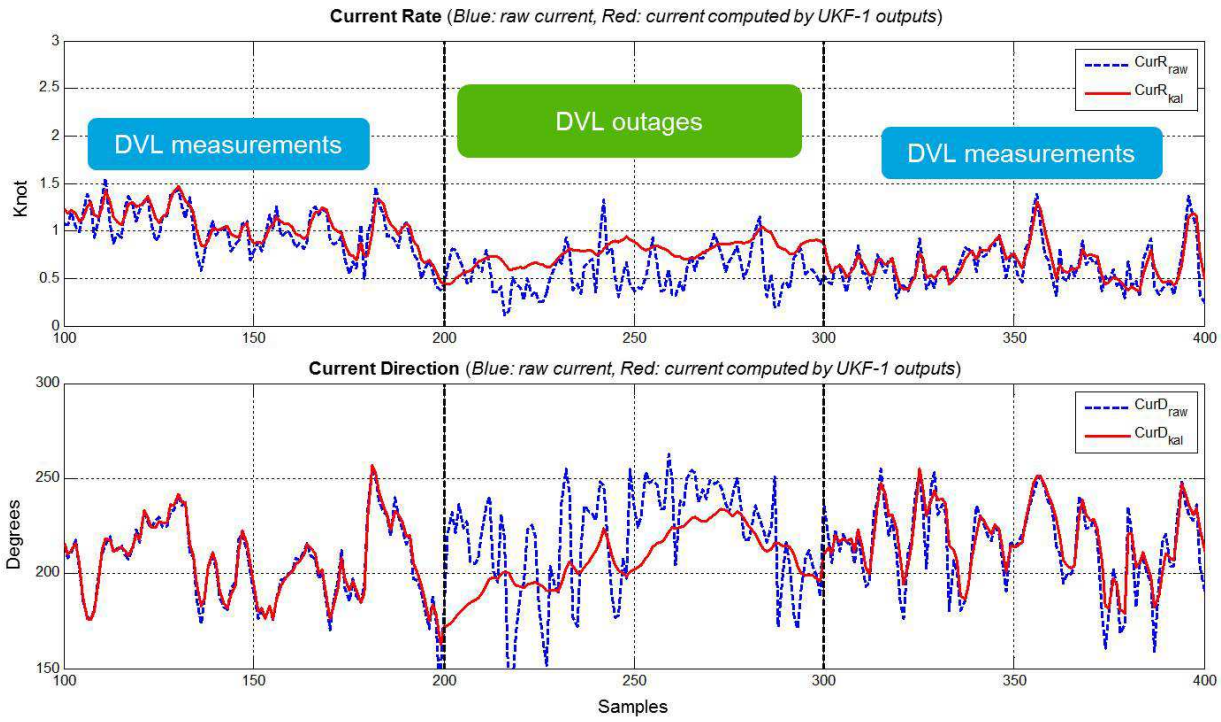


Figure 77. Current estimation of Model 3

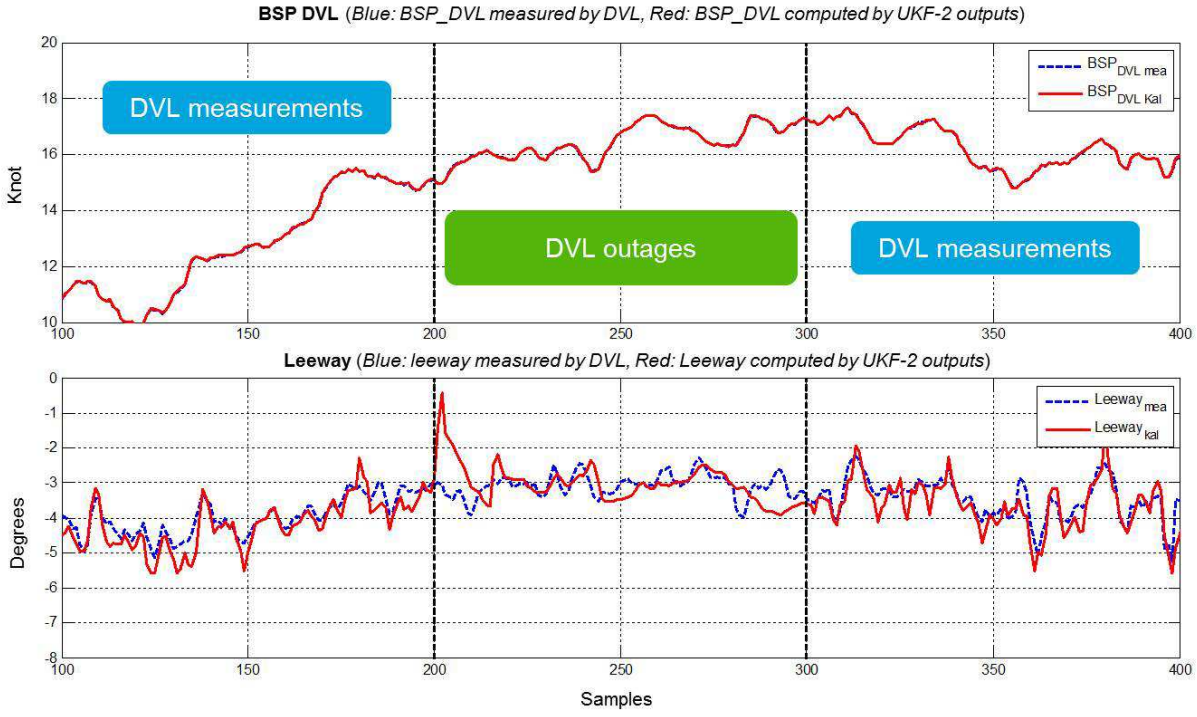


Figure 78. Leeway and BSP estimation of Model 3

Analysis

These three models respond differently. In the model 1, the current estimation is very smoothed, the drawback is a loss in term of tracking accuracy especially on leeway estimation. But the leeway prediction during DVL outages is correct. Unlike, this model, the leeway prediction on model 2 is unstable during the same period; these results show the limits of linearization by an EKF. In the model 3, we can see that the current estimation follow the raw measurements during the phase with DVL measurements and during the other phase we observe the leeway is not correct at initial DVL outage, but it converges over time.

Initially, these results highlight the imprecision of measurements and the interest to merge the DVL measurements with our model in order to improve the accuracy on leeway and ocean current estimation. So, from the results obtained, to linearize the system, the model based on the EKF is rejected since it is firstly complex to implement (Jacobian computation) and secondly appears to be quite unstable, or at least difficult to tune in order to reach stability. The KF model with pre-linearization is stable but also suffers from a relative lack of tracking precision. The UKF-based solution shows good properties in terms of tracking and stability but requires a delay to converge. Model-3 is the best individual solution but we can also consider a combination of model 1 and model 3 to improve tracking and convergence time in case of DVL outages. In all cases, we lack references on ocean current and leeway to fully validate this choice.

CONCLUSION

This paper gives an overview of the problems of the wind and boat displacement measurements with a particular attention to the leeway estimation error. Our first approach has shown that it is necessary to have sufficient accuracy on the leeway estimation in order that the system converges to the true values (current and leeway angle). But in fact, we do not know the error of this empirical formula of leeway; we think it is not accurate enough since it does not take into account new parameters such as keel angle or height daggerboard. That is why, we propose to use a new sensor, a DVL, in order to measure the surface speed and the leeway. Unfortunately, this sensor is also disturbed by several phenomena, so we offer a second approach based on a DVL sensor coupled with sophisticated signal processing. We demonstrate that a joint estimation of leeway and ocean current can be obtained with a closed-loop approach including cascaded filters. Our study shows that the UKF filter provide a good tracking/stability tradeoff and can be combined with a standard KF with pre-linearization in case of DVL outages. But DVL sensors are expensive and not all sailing team can afford such sensors. Based on our experiments we think that a solution is to use or rent a DVL during a measurement campaign in order to obtain an accurate leeway estimate. Then, using fitting techniques, we can believe that an ad hoc Leeway formula can be obtained while considering more parameters. Moreover, it would be interesting to compare DVL measurements with the boat performance, which are derived from the Velocity Prediction Program (VPP) that would allow improving or validating VPP program.

Our on-going work is the implementation of the proposed algorithms within on open and low-power navigation processor that sailing teams could use to develop and optimize their own algorithms.

ACKNOWLEDGEMENTS

We would like to thank the Team Sodebo which has provided the dataset with DVL measurements, we are especially grateful to Thomas Coville (skipper) for accepting the collaboration and Martin Gaveriaux (engineer) for data analysis support.

REFERENCES

J.Bijker, W. Steyn (2008), “Kalman filter configurations for a low-cost loosely integrated inertial navigation system on an airship”, Control Engineering Practice, Volume 16 (12), 1509-1518.

J.M. Bourgeon, S. Dyen, D Schmah, (2010), "l’Hydroptere: A story of a dream," 21st HISWA Symposium.

M. Brettle, (2001), "Wind Measurement," AWE International.

A. Gentry, (1981), "Sailboat Performance Testing Techniques," 11th AIAA Symposium on the Aero/hydrodynamics of Sailing, Seattle, Washington,.

A. Gentry, (2006), "The Origins of Lift," Sailing Technical Papers, Seattle, Washington.

M. Johnson, C. Hodgson, D. Gautier, (2011), “Method and device for determining wind conditions around a sailboat”.

S.J. Julier, J.K. Uhlmann, (1997), “A New Extension of the Kalman Filter to Nonlinear Systems”, the Proc of Aero Sense: 11th Int Symposium Aerospace/ Defense Sensing, Simulation and Controls.

R.E Kalman, (1960), “A New Approch to Linear Filtering and Prediction Problems”, Transaction of the ASME -Journal of Basic Engineering.

DJ Lee, (2005), "Nonlinear Bayesian filtering with application to estimation and navigation," Texas A&M University.

D. Pedrick and Richard McCurdy, (1981), "Yacht Performance Analysis with Computers," CSYS, Annapolis, MD.

A. Pons, D. Asiain, F. Quero, J. Cuevas, (2004), “Racing Bravo. Un sistema de navegacion para alta competicion”, Ingeniera Naval.

F. Rongere, J.-M. Kobus, (2010), “Mesure des trois composantes du vent sur support flottant avec correction des mouvements de plateforme”, Tech. Report, Laboratoire de Mécanique des Fluides de l’Ecole Centrale de Nantes.

E. Wan, R. Van Der Merwe, (2000), “*The Unscented Kalman Filter for Nonlinear Estimation*”, In Proceedings of IEEE Symposium on Adaptive Systems for Signal Processing Communications and Control.

G. Welch and G. Bishop, (2006), “An Introduction to the Kalman Filter”, Tech. Report TR 95-041, University of North Carolina at Chapel Hill.