Génération de modèles de haut niveau enrichis pour les systèmes hétérogènes et multiphysiques

L. Bousquet

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Nanoélectronique et nanotechnologies**
Arrêté ministériel : 7 août 2006

Présentée par

Laurent BOUSQUET

Thèse dirigée par **Emmanuel SIMEU** et
codirigée par **Katell MORIN-ALLORY** et **Laurent FESQUET**

préparée au sein du **Laboratoire TIMA – Techniques de l'Informatique et de la Microélectronique pour l'Architecture des systèmes intégrés**
dans **l'École Doctorale EEATS – Electronique, Electrotechnique, Automatique et Traitement du Signal**

Génération de modèles de haut niveau enrichis pour les systèmes hétérogènes et multiphysiques

Thèse soutenue publiquement le **«29 janvier 2014»**,
devant le jury composé de :

M. Patrick GIRARD
Directeur de Recherches au CNRS, LIRMM, Président

M. Fabrice MONTEIRO
Professeur des Universités, Université de Lorraine, Rapporteur

M. Ian O'CONNOR
Professeur des Universités, Ecole Centrale de Lyon, Rapporteur

Mme. Katell MORIN-ALLORY
Maître de Conférences, Institut Polytechnique de Grenoble, Co-encadrante

M. Laurent FESQUET
Maître de Conférences, Institut Polytechnique de Grenoble, Co-encadrant

M. Emmanuel SIMEU
Maître de Conférences, Université Joseph Fourier, Directeur de thèse

Remerciements

J’adresse tout d’abord mes sincères remerciements aux membres du jury pour le temps qu’ils ont consacrés à la lecture de ce mémoire et pour s’être déplacé pour la soutenance. Je remercie Monsieur Ian O’Connor, professeur à l’Ecole Centrale de Lyon et Monsieur Fabrice Monteiro, professeur à l’Université de Lorraine d’avoir accepté la tâche de rapporteurs. Je remercie également Monsieur Patrick Girard, directeur de recherches au CNRS d’avoir accepté de présider le jury.

Je tiens à exprimer toute ma gratitude à Monsieur Emmanuel Simeu, mon directeur de thèse durant ces quatre années. Je le remercie pour sa patience et ses conseils précieux. Je remercie aussi Madame Katell Morin-Allory et Monsieur Laurent Fesquet, mes co-encadrants.

Je remercie Monsieur Salvador Mir pour m’avoir accueilli au sein de l’équipe RMS, qu’il codirige avec Emmanuel. Je remercie également Madame Dominique Borrione, directrice du laboratoire TIMA pour la qualité de son accueil et pour m’avoir permis d’effectuer cette thèse dans d’excellentes conditions.

Je tiens à exprimer ma considération à deux anciens doctorants du laboratoire, Fabio Cenni et Franck Paugnat. Nos sujets de thèse étant basés sur SystemC AMS, nous avons été amenés à échanger, travailler puis publier ensemble. Mes premières publications ont été obtenues avec eux et je les remercie pour ces collaborations. Bien que n’ayant pas encore eu l’occasion de collaborer étroitement avec Torsten Mähne, je le remercie pour les quelques conversations que nous avons eu et pour les conseils qu’il a pu donner sur le forum internet de SystemC AMS : sa grande maitrise du langage m’a permis de mieux appréhender certains de ses concepts.

Je remercie tout le personnel de TIMA pour la bonne ambiance dans le laboratoire, en particulier les doctorants de l’équipe RMS pour les discussions, scientifiques ou non que nous avons pu avoir. Je pense à Yoann, Stéphane, Louay, Ke, Rafik, Nourredine, Rshdee, Brice, Asma, Martin et Guillaume. Je remercie tout particulièrement le personnel administratif : Anne-Laure, Laurence, Sophie, Youness et Nicolas, Ahmed et Frédéric du service informatique pour leur compétence. Je remercie également Alexandre Chagoya, du CIME Nanotech pour sa bonne humeur et sa disponibilité.

Il est aussi nécessaire de s’aérer l’esprit, ce que j’ai pu faire en pratiquant le tennis de table. Je remercie donc les gens qui ont fait partie de l’ancienne équipe de Donzère pour les matchs et les après matchs parfois épiques : Gillou, Dan, Ben, Cédric, Tony, Hermann, Seb et Fab. Merci aussi aux amis Donzérois de longue date : Marie, Matt, Romain et Jo !

Enfin, je remercie mes parents qui m’ont toujours soutenu. Je leur dédie ce mémoire.

Table des matières

Index des Figures

Index des Tableaux

1. Introduction

Un système embarqué est un système complexe qui intègre du logiciel et du matériel conçus ensemble afin de fournir des fonctionnalités données. Il contient généralement un ou plusieurs microprocesseurs destinés à exécuter un ensemble de programmes définis lors de la conception et stockés dans des mémoires. Le système matériel et l'application (logiciel) sont intimement liés et immergés dans le matériel et ne sont pas facilement discernables, comme dans un environnement de travail classique de type ordinateur de bureau. Les systèmes embarqués occupent une place prépondérante dans nos vies, nous les utilisons quotidiennement. Depuis une quinzaine d'année et la démocratisation de l'usage de la téléphonie mobile, chacun dispose d'au moins un système embarqué dans sa poche. Ceci est rendu possible par le transistor, que l'on peut qualifier, sans prendre de risques, d'objet technique du vingtième siècle. Inventé en 1947 par Shockley, Bardeen et Brattain, le transistor a pu, en à peine soixante ans, être intégré par milliards sur un cm^2 par les industriels. Cette miniaturisation entraine une complexification des systèmes embarqués qui ne sont plus de simples unités de calculs. Le cœur du système embarqué ou unité de calcul demeure de nature numérique, il peut être matériel ou logiciel. Il reçoit des informations provenant du monde extérieur par l'intermédiaire de capteurs. Ensuite ces informations sont traitées, ceci permettant de commander des actionneurs. Les capteurs et les actionneurs peuvent eux-mêmes évoluer dans différents domaines de la physique tels que la mécanique, la chimie, l'optique ou l'hydraulique. Les interfaces entre le monde physique extérieur et le cœur du système sont de nature analogique. Il s'agit donc de systèmes hétérogènes dont la conception, de plus en plus complexe, demande des compétences très variées.

Parallèlement, une concurrence importante s'est établie entre les entreprises du secteur. Elles doivent être capables de produire rapidement et avec des coûts les plus réduits possibles. Le temps de développement doit donc être réduit afin d'accélérer la mise sur le marché des produits. Or, il est très difficile d'appréhender des systèmes aussi complexes dans leur globalité tout en maintenant beaucoup de détails. Une solution consiste à utiliser des modèles de haut niveau d'abstraction afin de pouvoir réaliser des simulations globales dans des temps raisonnables.

D'une façon générique, l'architecture d'un système embarqué se compose d'une partie logicielle, qui est implantée sur une plate-forme matérielle. Ainsi, le développement et la validation du logiciel embarqué requièrent la disponibilité d'un prototype complet de la plate-forme matérielle d'implantation. Afin de gagner du temps, un modèle informatique de la plate-forme matérielle est généralement utilisé. Ce modèle est appelé prototype virtuel. Le logiciel embarqué peut ainsi être développé puis testé sur un prototype virtuel avant que la plate-forme matérielle ne soit disponible. Les interactions entre la partie logicielle et la partie qui sera *in fine* matérielle peuvent être modélisées. Ce travail de prototypage virtuel est souvent réalisé dans l'environnement de programmation SystemC.

SystemC permet de réaliser efficacement le prototypage virtuel de la partie matérielle numérique d'un système embarqué, puis le développement de la partie logicielle, mais il ne permet pas de modéliser correctement les systèmes à signaux mixtes et analogiques (Analog and Mixed-Signal – AMS). Pour être certain du fonctionnement futur d'un système, il est nécessaire de pouvoir effectuer des simulations globales, une analyse partitionnée n'étant pas suffisamment fiable. Des simulations conjointes (ou cosimulations) permettent de réaliser ces simulations globales requises, mais elles reposent généralement sur des outils de conception assistée par ordinateur (CAO) et des plates-formes logicielles différentes. Outre le fait que la synchronisation de ces outils peut s'avérer délicate, cette solution est coûteuse en termes de durée de simulation. Un autre inconvénient est que le simulateur électrique, souvent de type SPICE, utilise des modèles très détaillés, basés sur un grand nombre de paramètres. Cela introduit un grand nombre d'équations à résoudre par le solveur et donc des durées de simulation importantes. Les simulations de modèles créés à partir des langages de description matérielle pour l'analogique, bien que plus rapides que les simulations SPICE demeurent elles aussi trop lentes et ne conviennent pas non plus. Il faut donc être capable d'élever le niveau d'abstraction des parties AMS. Afin de pallier les inconvénients de la cosimulation, il faut disposer d'outils qui soient capables de modéliser et simuler un système embarqué hétérogène dans sa globalité, dans un environnement logiciel unique. Ce second point n'implique pas obligatoirement de créer un outil complètement nouveau. Il peut s'agir de proposer une extension à un outil existant et ayant fait ses preuves sur la partie numérique. C'est le choix qui a été retenu : produire une extension de SystemC pour l'analogique et les signaux mixtes : SystemC AMS. Une fois le système complet correctement modélisé à haut niveau, il faut disposer d'une méthodologie pour les parties analogiques qui permette d'obtenir finalement le modèle très détaillé correspondant au système réel. Cela doit s'apparenter au flot de conception numérique, c'est-à-dire une approche descendante de raffinements successifs, aussi connue sous le nom d'approche top-down. Ainsi, une méthodologie top-down, analogue au flot de conception numérique, est à définir pour les parties analogiques.

Pour relever ces défis, différents acteurs majeurs du secteur de l'électronique sur le continent européen se sont rassemblés au sein du projet B-DREAMS (Beyond Design Refinement of Embedded Analogue and Mixed-signal Systems). Ce projet européen a été financé par le groupement CATRENE (Cluster for Application and Technology Research in Europe on NanoElectronics) et entre dans le cadre du programme Eurêka. Il s'est déroulé d'octobre 2008 à octobre 2011. L'objectif est d'assurer aux industries européennes une position dominante dans le domaine de la conception des systèmes embarqués à signaux analogiques et mixtes (Embedded Analog Mixed-Signal systems – EAMS systems). Afin de rester compétitif, il est nécessaire de réduire le temps de mise sur le marché tout en maitrisant les coûts de développement et de production. Les principaux objectifs de B-DREAMS consistaient à :

- Etablir un cadre pour la modélisation et la simulation des prototypes virtuels AMS.
- Fournir une méthodologie de raffinement des modèles AMS.
- Fournir une méthodologie permettant la réutilisation des blocs matériels propriétaires analogiques (Analog Intellectual Property – aIP).
- Permettre la modélisation, la simulation et la vérification des parties analogiques conjointement avec les parties numériques.
- Faire des propositions devant conduire à la normalisation de SystemC AMS par l'OSCI (Open SystemC Initiative).
- Faire des propositions devant consuire à la normalisation de l'extension de IP-XACT à l'analogique par Accellera.

La Figure 1 fait apparaître à gauche deux niveaux d'abstraction manquants. A droite apparaissent les projets B-DREAMS, SYENA et CAIRO. SYENA est un projet du Ministère Fédéral Allemand de l'Éducation et de la Recherche et CAIRO est un projet du Laboratoire d'Informatique de Paris 6 (Université Pierre et Marie Curie). Ces deux projets étaient axés sur la modélisation analogique de bas niveau et sur la synthèse analogique au moyen de blocs matériels analogiques. Les acteurs de ces projets ont été intégrés au projet B-DREAMS.

Le projet B-DREAMS a impliqué des grandes entreprises, des PME, des centres de recherche et de transfert de technologie ainsi que des institutions académiques :

- Des entreprises de l'industrie des semiconducteurs :
  - Infineon – Allemagne
  - NXP Semiconductors – Pays-Bas
  - STEricsson – France
  - STMicroelectronics – France
- Un équipementier spécialiste de l'électronique pour l'automobile :
  - Robert Bosch Automotive Electronics - Allemagne
- Des PME spécialisées dans les outils de CAO :
  - Dizain-Sync – Pays-Bas
  - Magillem Design services – France
  - Twente Institute for Wireless and Mobile Communications – Pays-Bas
- Des centres de recherche et de transfert de technologies :
  - CEA-LETI – France
  - Fraunhofer IIS/EAS (FhG) – Allemagne
  - IMEC-NL – Pays-Bas
- Des institutions académiques :
  - Institut Polytechnique de Grenoble, Laboratoire TIMA – France
  - Université Pierre et Marie Curie, Laboratoire LIP6 – France
  - Université Technologique de Delft – Pays-Bas
  - Université Technologique de Vienne – Autriche

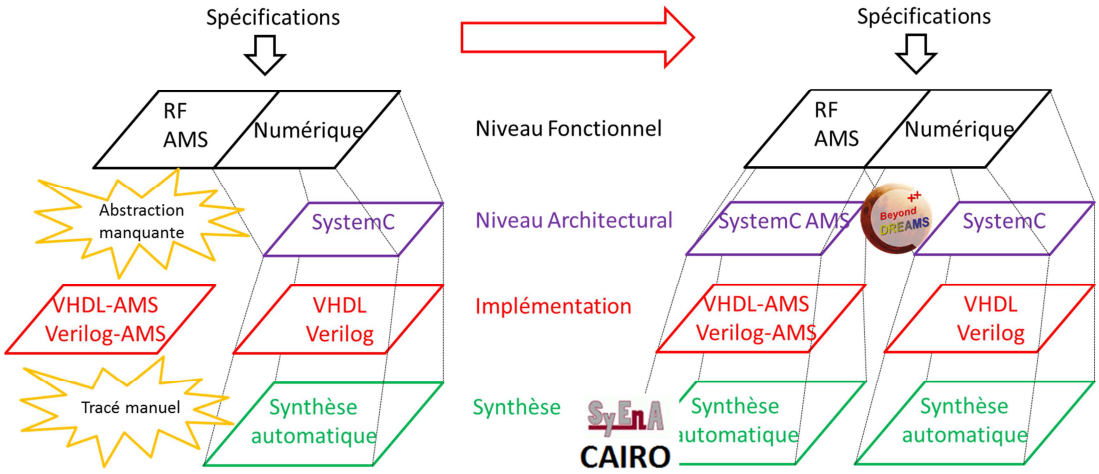


Figure 1. Positionnement de B-DREAMS par rapport au flot AMS

Au sein du laboratoire TIMA, deux thèses se sont déroulées précédemment dans le cadre du projet B-DREAMS. La thèse de Fabio Cenni, qui s'est déroulée en collaboration avec STMicroelectronics a essentiellement porté sur le prototypage virtuel d'un capteur d'image CMOS à différents niveaux d'abstraction. Ce travail a permis de débuter la phase de développement du logiciel neuf mois avant la réception du capteur. D'autres travaux portant sur la génération de modèles de haut niveau ont aussi été menés. En se basant sur la courbe de réponse fréquentielle, des techniques d'ajustement analytique permettent soit de réduire l'ordre du modèle, soit d'identifier des modèles à partir d'analyses menées au moyen d'autres outils de conception que SystemC AMS. Ce flot apparaît en vert sur la Figure 2. Une autre méthode étudiée permet de générer des modèles « boîtes noires » à partir de données empiriques ; qu'elles soient mesurées sur un système physique, simulées à partir d'un modèle informatique ou provenant d'une autre source. Ce flot apparaît en rouge sur la Figure 2. La thèse de Franck Paugnat s'est focalisée sur deux principaux axes. Le premier a consisté à proposer une méthodologie de conception et de raffinement pour les systèmes AMS et à définir la place de SystemC AMS dans ce flot. Le second axe présente un modèle SystemC AMS d'amplificateur opérationnel simplifié.

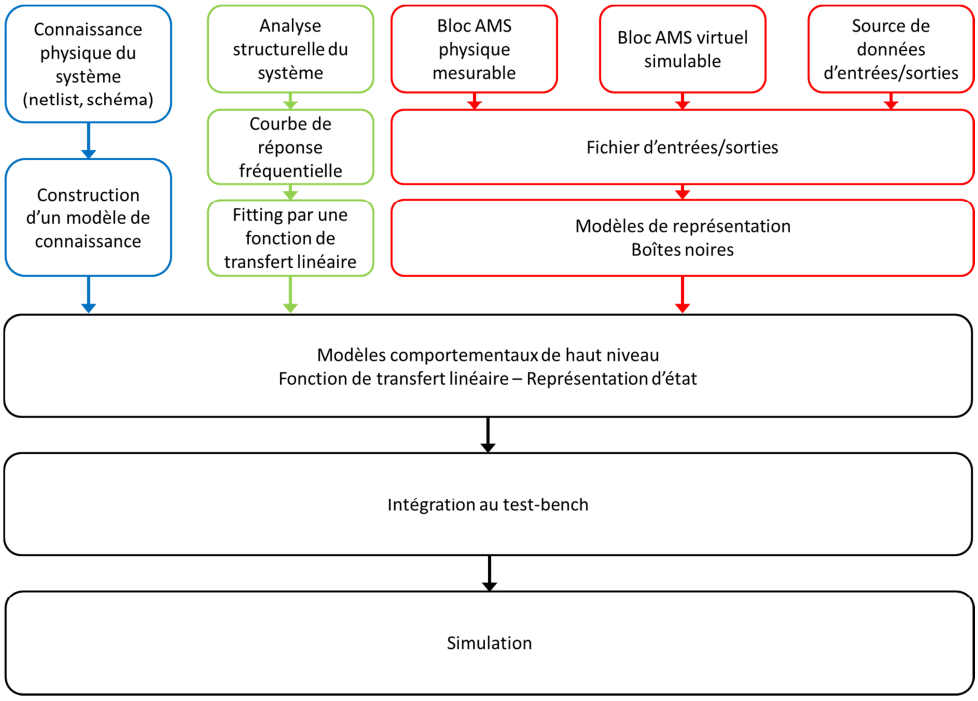


Figure 2. Travaux menés à TIMA sur la modélisation AMS à haut niveau

Cette thèse s'inscrit elle aussi dans le cadre du projet B-DREAMS. Le travail s'est essentiellement focalisé sur la génération automatique de modèles de connaissance. Si l'on se réfère à la Figure 2, ceci correspond au flot apparaissant en bleu. Le chapitre 3 présente des travaux menés conjointement avec Franck Paugnat sur les modèles de calcul de SystemC AMS et les différents styles de modélisation possibles. Nous introduisons notamment un style de modélisation original établi par une combinaison de deux modèles de calculs existant ainsi qu'une bibliothèque permettant de simplifier l'utilisation de ce style de modélisation. L'intérêt de ce style de modélisation est réduit depuis la parution de la version 1.0 de SystemC AMS mais il conserve toutefois certains attraits. Deux cas d'étude sont ensuite présentés afin d'établir un lien entre le type et la taille du modèle et la durée de simulation. Le chapitre 4 présente comment établir un modèle de connaissance d'un circuit analogique linéaire. Tout d'abord, une méthode d'abstraction est présentée : à partir de la netlist d'un circuit linéaire, la représentation d'état est générée automatiquement. Ceci pouvant s'inscrire dans le cadre de l'exploration architecturale, nous présentons ensuite un flot de raffinement pour les filtres analogiques : à partir des spécifications, un circuit est automatiquement généré. Enfin, nous montrons comment il est possible d'enrichir le modèle de haut niveau avec des informations permettant de calculer la consommation du circuit. Différents exemples sont traités.

Le chapitre 5 propose d'appliquer les méthodes précédentes aux systèmes électromécaniques. La partie mécanique est ramenée à son équivalent électrique, la méthode d'abstraction pouvant alors s'appliquer. Le chapitre 6 propose de traiter le problème des systèmes électromécaniques au moyen des bond graphs. Une méthode automatique permettant de générer un bond graph à partir d'une description de bas niveau est proposée. Le chapitre 7 présente le prototype virtuel d'une éolienne. Les différents concepts vus jusqu'alors y sont illustrés. Le propos s'achèvera par les chapitres 8 et 9 où des perspectives de travaux futurs et d'axes d'améliorations possibles puis les conclusions des travaux réalisés seront donnés. Mais afin de mieux comprendre le contexte ayant conduit le développement de SystemC AMS, le chapitre 2 présentera un historique des outils et méthodes de modélisation et de conception des systèmes AMS puis décrira brièvement les travaux reposant sur SystemC AMS qui ont été menés jusqu'à présent.

2. Modélisation et conception des systèmes hétérogènes

Ce chapitre est une présentation générale des systèmes hétérogènes et des moyens qui sont utilisés pour les modéliser. Nous définirons tout d'abord les différents types d'hétérogénéités que l'on peut rencontrer dans ces systèmes, ainsi que leur flot de conception et leur modélisation. Puis, nous aborderons les outils et techniques disponibles pour la modélisation, la simulation et la conception. Une brève introduction de SystemC et de SystemC AMS sera donnée. Enfin, un tour d'horizon des travaux marquants portant sur la modélisation des systèmes hétérogènes au moyen de cette plate-forme sera donné.

2.1 Les systèmes hétérogènes

Dans cette partie, nous allons définir les hétérogénéités de type matériel/logiciel, puis au sein de la partie matérielle nous verrons que l'on peut rencontrer une hétérogénéité de type analogique/numérique. Le dernier type d'hétérogénéité que nous aborderons sera le cas des systèmes multiphysiques.

2.1.1 Matériel et logiciel

Le premier type d'hétérogénéité peut se trouver au cœur même du système embarqué : tels ou tels calculs ou tâches seront effectuées soit par un module matériel dédié soit par un programme (software, SW) exécuté sur un microprocesseur. Si l'on a affaire à une tâche critique ou que l'on doit réaliser de très grandes séries, la meilleure solution est d'opter pour un circuit intégré dédié (ASIC) mais ceci a un coût. Une autre solution matérielle consiste à utiliser un circuit configurable (FPGA).

2.1.2 Hétérogénéité électrique

On parle d'hétérogénéité électrique lorsque, au sein d'un système coexistent des parties purement numériques et des parties analogiques. Les signaux analogiques peuvent se situer en entrée du système, par exemple lorsque le système doit traiter des signaux provenant de capteurs. Ces signaux sont analogiques car ils correspondent à des grandeurs physiques. Ainsi, le signal de sortie d'un capteur de température pourra évoluer sur une certaine plage de tension. Dans la plupart des cas de systèmes complexes, il devra être converti en valeur numérique pour être par la suite traité par différentes fonctions du système. De même, en sortie du système, si l'on doit piloter un moteur par une tension ou une fréquence, on sera aussi en présence de signaux analogiques, la fréquence ou la tension du signal de contrôle du moteur pourront évoluer sur une certaine plage. Le cœur du système utilise, lui, des signaux électriques qui ne peuvent prendre que deux valeurs différentes '0' ou '1'. En fait, dès qu'un système embarqué doit interagir avec le monde extérieur, qui lui n'est pas binaire, il devra intégrer des composantes numériques et analogiques. Bien sûr, il intégrera presque toujours les composants capables d'assurer l'interaction entre les deux parties : les convertisseurs analogique vers numérique (ADC) et numérique vers analogique (DAC). Malgré tout, on se situe dans un et unique domaine de la physique : l'électricité.

2.1.3 Hétérogénéité multiphysique

Dans les stratégies de conception actuelles, un système embarqué est généralement intégré sur un support silicium unique constituant ainsi un système complet intégré sur une puce (System on Chip, SoC). Les systèmes sur puce contiennent généralement une grande variété de dispositifs programmables qui cohabitent sur le même support de silicium avec des composants analogiques et mixtes divers tels que des composantes radiofréquences (RF) comme moyen de communication, des composantes optiques pour le transfert de données à haut débit, des systèmes microélectromécaniques (MEMS) pour l'interfaçage avec le monde externe ou des convertisseurs analogique/numérique et numérique/analogique requis pour le dialogue interne. Ces systèmes peuvent intégrer les capteurs directement sur le silicium de la puce. Cette nouvelle stratégie de conception oblige les concepteurs à prendre en compte des contraintes qui ne les préoccupaient pas par le passé. L'objectif est d'obtenir une coopération harmonieuse entre composants embarqués afin de garantir des services globaux. Dans le paragraphe précédent, nous avons abordé un type d'hétérogénéité découlant de la différence de nature des signaux électriques, les signaux analogiques étant générés par des capteurs physiques ou par le système embarqué en vue de piloter des actionneurs. Si l'on désire modéliser et simuler le système sur puce dans son ensemble, on devra aussi considérer des grandeurs physiques de natures différentes telles que la température, la force mécanique, les concentrations d'éléments chimiques, etc.

2.1.4 Résumé

Comme nous l'avons vu dans les paragraphes précédents, il peut exister plusieurs types d'hétérogénéités au sein d'un système embarqué. La Figure 3 illustre les différents cas. La zone orange représente le domaine de l'électronique numérique. A l'intérieur de cette zone, on peut rencontrer l'hétérogénéité matérielle/logicielle : certaines fonctions pouvant être assurées par un programme alors que d'autres seront assurées par un circuit spécifique. La zone bleue représente le domaine de l'électronique analogique, des convertisseurs sont à l'interface avec le domaine électronique numérique et c'est à ce niveau que se trouve l'hétérogénéité électrique. Enfin, la zone grise représente le monde extérieur et l'interaction avec le domaine électrique analogique est assurée *via* les capteurs ou les actionneurs. C'est ici qu'apparaît l'hétérogénéité multiphysique. Dans certains cas, la partie numérique peut directement piloter des actionneurs s'ils ne nécessitent qu'une faible puissance pour fonctionner, comme les moteurs pas à pas par exemple. Par la suite, nous ne nous intéresserons qu'aux hétérogénéités de type électrique et multiphysique et lorsque nous mentionnerons les systèmes à signaux mixtes c'est à ce type d'hétérogénéités que nous nous référerons.

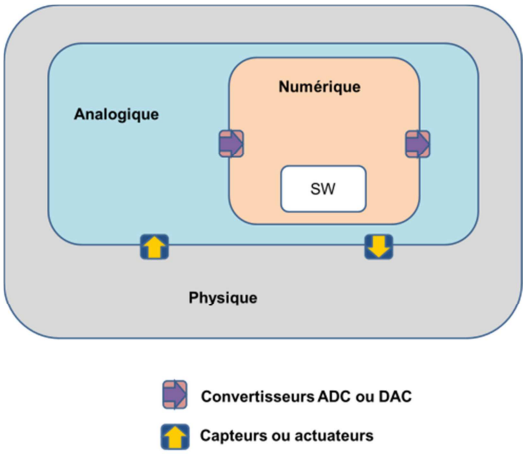


Figure 3. Vue d'ensemble d'un système embarqué hétérogène

2.2 Conception des systèmes hétérogènes

Pour illustrer le processus de conception des systèmes hétérogènes, nous considérons les différentes phases de la conception des MEMS. En partant des spécifications, le concepteur essaie tout d'abord de trouver un composant correspondant, s'il n'en trouve pas, alors des modifications seront nécessaires. Traditionnellement, deux équipes d'ingénierie collaborent à la conception de MEMS. La première utilise un outil d'aide à la conception (CAD) d'Analyse par Eléments Finis (FEA) comme CoventorWare [Khankhua 11] ou ANSYS [Malik 08] pour créer un modèle électromécanique. La seconde équipe utilise un outil Automatique de Conception Electronique (EDA) tel que Cadence, MentorGraphics, ou Synopsys afin de réaliser des simulations au niveau transistor. Pour vérifier l'interfaçage des aspects mécaniques et électriques, des modèles comportementaux du MEMS sont développés en utilisant des Langages de Description Matérielle Analogique (AHDL), par exemple VHDL-AMS, en se basant sur le comportement de la caractérisation par éléments finis. Cela permet de prendre en compte le couplage entre les domaines mécaniques et électrostatiques se produisant au niveau du MEMS ainsi que les circuits numériques et analogiques de contrôle et de traitement. Le layout de la structure mécanique et du circuit électronique est produit par un outil de layout. Ainsi, des outils de natures différentes sont nécessaires à ce développement, ce qui ne permet pas de vraiment optimiser les systèmes résultants. Pour des systèmes incorporant des MEMS, les méthodes de conception existantes ne sont pas efficientes, elles nécessitent des itérations et sont donc coûteuses en temps. Le flot de conception des parties analogiques ou RF présente des difficultés similaires, à ceci près que les outils d'Analyse par Eléments Finis sont remplacés par des simulateurs spécifiques comme SPICE ou Agilent ADS. Les langages de description analogique sont néanmoins utilisés pour simuler l'électronique environnante puisqu'ils permettent de réduire les durées de simulation en faisant intervenir moins de détails dans les modèles.

La description précédente permet de cerner le principal problème actuel dans le flot de conception des systèmes hétérogènes : différentes disciplines interviennent, faisant chacune appel à des spécialistes et des outils spécifiques : le flot de conception se sépare en plusieurs branches. Un moyen d'unifier le flot de conception entre le niveau architectural et le niveau implémentation est nécessaire. Dans le même esprit, il manque une méthodologie et un langage associé permettant de valider les systèmes hétérogènes au niveau application. Un langage de haut niveau est nécessaire pour modéliser la partie logicielle et les parties matérielles numériques et analogiques au moyen de Modèles de Calculs (MoCs) dédiés. La synthèse dans les domaines AMS n'étant toujours pas réalisable, bien que des travaux soient en cours dans ce sens [der Plas 02], [O'Connor 06], [Mitea 11], la conception des systèmes AMS devrait être basée sur le raffinement de modèles validés par différents outils, à différents niveaux d'abstraction, et une extension de SystemC à l'analogique et aux signaux mixtes devrait jouer un rôle clé à l'avenir dans la modélisation des systèmes hétérogènes à haut niveau d'abstraction.

2.3 Raffinement des systèmes hétérogènes basé sur la modélisation

Dans le cas idéal, l'approche top-down ou raffinement du haut niveau vers le bas niveau d'abstraction (ou encore du niveau de détails le plus faible vers le niveau de détails le plus élevé) débute par une spécification exécutable du comportement attendu au niveau système. Le processus de raffinement est une approche pas à pas consistant à remplacer les blocs du niveau système par des modèles plus précis, ou moins abstraits. En partant des spécifications globales, des unités fonctionnelles sont identifiées puis séparées et divisées en blocs architecturaux. C'est à ce niveau que les extensions de SystemC vont intervenir : à un niveau d'abstraction plus élevé que celui offert par VHDL-AMS, comme l'illustre la Figure 4. Bien sûr, le paradigme top-down n'est pas le seul paradigme possible pour la conception des circuits intégrés, mais c'est le plus évident. Dans la vie, lorsque nous arrivons sur un lieu inconnu, nous avons tout d'abord une vue d'ensemble, puis nous essayons d'analyser les détails qui s'offrent à nous.

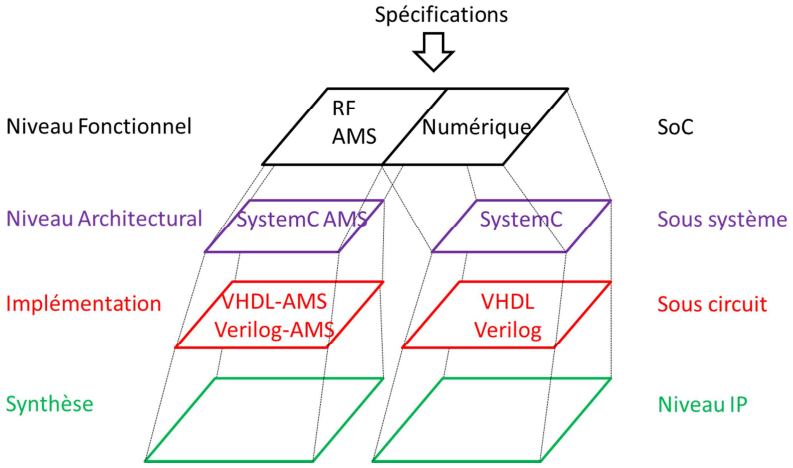


Figure 4. Les niveaux d'abstractions d'un système hétérogène

2.4 Les outils et langages de modélisation des systèmes hétérogènes

L'approche des systèmes hétérogènes multiphysiques a d'abord été compartimentée. Chaque domaine de la physique a ses propres outils de CAO. Chaque partie d'un système était modélisée indépendamment et par des spécialistes. Ainsi, les interactions n'étaient pas ou peu prises en compte. Outre le fait qu'elles peuvent avoir un impact important sur le comportement d'un système, les interconnexions des différentes parties ne pouvaient pas être vérifiées. De nombreuses erreurs ont été causées par cette impossibilité : inversion de polarité d'un signal, terminaux non connectés, connexions interverties…Ceci entrainant des retards de production et donc un coût non négligeable [Karnane 09].

Pour présenter les outils et langages permettant la modélisation des hétérogénéités, il faut commencer par Ptolemy II [Brooks 10]. Dans l'approche de Ptolemy, il est possible d'utiliser différents modèles de calcul afin de créer des modèles qui seront ensuite assemblés à différents niveaux hiérarchiques pour obtenir un modèle final complexe. Un modèle de calcul définit un formalisme de modélisation : un ensemble de règles syntaxiques et un jeu de calculs le définisse. Ces modèles peuvent être utilisés en simulation, synthèse ou preuve formelle et sont donc aussi appelés modèles exécutables. Parmi les modèles de calcul supportés par Ptolemy II, on trouve :

- Discrete Event MoC (DE) : il est basé sur une représentation discrète du temps, il convient donc de l'utiliser pour modéliser des systèmes numériques ou échantillonnés. Ses règles de calcul sont définies formellement par un simulateur à logique évènementielle.
- Continuous Time MoC (CT) : il est basé sur une représentation continue du temps et convient donc pour la modélisation des systèmes analogiques ou RF. Les règles de calculs sont définies par un solveur d'équations ou un simulateur de circuit.
- Synchronous Data Flow MoC (SDF) : c'est un MoC asynchrone dans lequel des piles First-In, First-Out (FIFO) modélisent les communications entre les processus. Les

processus encapsulent les comportements séquentiels et les états possibles. Ce MoC convient pour la modélisation d'applications de traitement du signal.

L'environnement Ptolemy II a été considérablement utilisé dans le cadre de travaux de recherche et a inspiré le développement d'autres plates-formes de conception telles que Metropolis [Balarin 03] ou ForSyDe [Sander 04]. Leur principal avantage sur Ptolemy II est qu'elles fournissent des méthodes formelles depuis les spécifications jusqu'à l'implémentation grâce au raffinement formel ou à des étapes de synthèse. Le principal inconvénient de la plate-forme Ptolemy II est son implémentation utilisant la structure Java, qui s'avère lente si des systèmes complexes doivent être simulés. Ptolemy II n'est pas un langage à proprement parler ; le choix retenu ici est d'utiliser les méthodes et techniques spécifiques de chaque domaine physique. Les différents MoCs ou l'interconnexion à des outils de conception spécifiques permettent de mettre en œuvre la cosimulation. Ptolemy II a prouvé son efficacité dans la création des prototypes virtuels hétérogènes. Cependant, il n'est pas possible de récupérer à haut niveau des informations de bas niveau obtenues après le raffinement de certaines parties [Agilent 04].

Pour ces raisons, la cosimulation est devenue indispensable. Le simulateur de circuits électriques SPICE a été amélioré, il est désormais plus rapide. Les simulateurs de type FAST SPICE permettent de répartir la simulation d'un schéma transistor sur plusieurs instances du simulateur, ces dernières pouvant même être exécutées sur plusieurs processeurs. Certes, cela entraine une perte de précision (due aux hypothèses faites pour répartir le circuit) de l'ordre de 5%, mais les simulations sont cinq à cinquante fois plus rapides [Goering 07][Santarini 08]. Toutefois, ce gain est faible par rapport aux résultats obtenus au moyen d'une simulation au niveau système. Cependant, l'association d'un simulateur numérique exécutant des modèles SystemC ou SystemVerilog avec un simulateur FAST SPICE ne permet pas d'atteindre des vitesses de simulation satisfaisantes. Malgré leur précision, les simulateurs électriques sont trop lents et pénalisent trop la cosimulation.

A un niveau d'abstraction plus élevé que celui de SPICE, VHDL-AMS et Verilog-AMS permettent de prendre en compte les hétérogénéités. Sur des modèles purement électriques, grâce à la modélisation comportementale, les durées de simulation VHDL-AMS ou Verilog-AMS sont environ cent fois plus faibles que celles de SPICE. Cependant, dans le cas de systèmes comportant une partie logicielle importante, comme un réseau de capteurs, le gain en vitesse de simulation est réduit [Vasilevski 08c]. D'où la nécessité d'élever encore plus le niveau d'abstraction des parties non numériques afin de pouvoir simuler des systèmes hétérogènes dans leur ensemble à des vitesses de simulation comparables à celle de SystemC TLM par exemple (nous reviendrons plus loin sur SystemC TLM).

Dans la modélisation par valeurs réelles RVM (Real Value Modeling), le temps est discrétisé mais pas l'amplitude, les signaux sont échantillonnés : les valeurs analogiques sont transcrites en valeurs réelles. La simulation évènementielle est effectuée par un simulateur numérique, la

vitesse est très rapide : de l'ordre de la vitesse de simulation d'une partie numérique. La modélisation RVM reste cependant imprécise : les valeurs analogiques n'étant mise à jour que lors d'un évènement numérique. Toutefois, elle peut être utilisée lorsqu'une grande précision n'est pas requise, lorsqu'on veut vérifier un comportement global, par exemple [Karnane 09][Hartong 09].

MATLAB/Simulink est un outil de simulation à haut niveau d'abstraction. De nombreuses fonctions permettant de modéliser des systèmes hétérogènes sont prédéfinies dans les bibliothèques disponibles. Cet outil est fréquemment utilisé pour comparer des résultats ou des durées de simulations avec ceux obtenus au moyen d'autres outils. Malgré tout, MATLAB est un outil mathématique, il n'a pas été conçu pour développer des programmes informatiques. C'est pour cela qu'il ne peut pas être l'unique outil de modélisation d'un système hétérogène. Il faudra l'utiliser conjointement avec un autre outil tel que SystemC par exemple. Toutefois, l'interconnexion des outils est une tâche difficile (synchronisation, échange de données…) et même en cas de réussite sur ce point, le coût en temps de calcul est élevé.

MASCOT [Bjureus 01] a été développé en tant que technique de modélisation intégrant Matlab [Matlab 11] et des modèles SDL (Specification and Description Language) dans un environnement de cosimulation. Il supporte aussi une méthodologie de modélisation, dans laquelle les spécifications sont abstraites dans un ensemble de processus communicant par le biais de FIFO de longueur infinie. Là encore, l'interconnexion des outils est coûteuse en temps de simulation.

Un tour d'horizon des méthodes et outils développés à l'Université Catholique de Louvain (KUL) pour la conception des circuits intégrés et des systèmes à signaux mixtes ou RF est présenté dans [Gielen 05]. Le besoin de disposer d'une bibliothèque riche de modèles comportementaux de base pouvant être utilisés à différents niveaux d'abstraction ou sur différents MoCs y apparaît clairement. Les auteurs soulignent aussi le fait que l'on manque de méthodes permettant de générer de façon systématique des modèles comportementaux de blocs analogiques ou RF.

D'autre part, des méthodes automatiques ont été développées pour l'extraction ou l'abstraction de modèles à partir de netlists [Nathke 04]. De telles approches s'accordent bien avec la vérification bottom-up ou approche ascendante du bas niveau d'abstraction vers le haut niveau d'abstraction (ou encore du niveau de détails le plus élevé vers le niveau de détails le plus faible) puisque les modèles ainsi générés représentent les comportements de circuits réels.

Parallèlement à ces avancées sur les outils de modélisation, des travaux ont été mené sur le développement des langages de modélisation pour la conception matérielle. Dans les années 80, les netlists au niveau transistor ou porte logique étaient utilisées pour la simulation de circuits ou la vérification LVS (Layout Vs Schematic). Dans les années 90, les langages de

description matérielle (HDL) comme VHDL [IEEE 09a] ou Verilog [IEEE 04] devinrent les principaux langages de conception pour la partie matérielle numérique. Des extensions pour l'analogique et les signaux mixtes ont aussi été développées, VHDL-AMS [IEEE 07] et Verilog-AMS [ACCELLERA 09]. Les années 2000 ont vu l'apparition des langages HDL basés sur C/C++ : SystemC [Black 10] [Grötker 02] [IEEE 06] [Müller 03], SpecC [Gajski 00] [Gerstlauer 01], Handel-C [Bowen 11] [Mentor 11] ont permis de développer des modèles plus abstraits et de pouvoir inclure les parties logicielles.

Le Tableau 1, d'après [O'Connor 07] montre les langages utilisés en fonction de la tâche à effectuer ou le niveau d'abstraction auquel on souhaite travailler.

Tableau 1. Langages utilisés en fonction du niveau d'abstraction et du domaine physique

<table>
<tr><th>Domaine Physique / Niveau d'abstraction</th><th>Logiciel</th><th>Numérique</th><th>Analogique</th><th>RF</th><th>Mécanique</th><th>Optique</th><th>Fluidique</th><th>Chimique</th></tr>
<tr><td>Service</td><td colspan="2">CORBA</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Transaction</td><td colspan="2">SystemC TLM/UML</td><td colspan="2">SystemC AMS</td><td colspan="4">MoCs spécifiques SystemC AMS</td></tr>
<tr><td>Macro-architecture</td><td colspan="2">SystemC</td><td colspan="3" rowspan="2">Ptolemy</td><td rowspan="2">SystemC</td><td></td><td></td></tr>
<tr><td>Micro-architecture</td><td colspan="2">SystemC/VHDL</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Bloc</td><td rowspan="3"></td><td>VHDL</td><td>VHDL-AMS</td><td>RF/VHDL-AMS</td><td rowspan="2">VHDL-AMS</td><td rowspan="2">VHDL-AMS</td><td>VHDL-AMS</td><td></td></tr>
<tr><td>Circuit</td><td>Simulation électrique</td><td>VHDL-AMS</td><td>RF</td><td>FEMLab</td><td></td></tr>
<tr><td>Physique</td><td colspan="4">Méthodes par éléments finis</td><td></td><td></td><td>analytique</td></tr>
</table>

Parallèlement à SystemC, SystemVerilog [IEEE 05] [IEEE 09b] est devenu un langage de spécification, description et vérification très utilisé. Il permet de créer des modèles de systèmes complexes au niveau des transferts entre les registres (RTL) et au niveau architectural et de faciliter leur vérification en utilisant les méthodes formelles. SystemVerilog se concentre sur les parties numériques et ne dispose pas d'extension au domaine AMS. Le groupe de travail Accellera Verilog Analog Mixed-AMS essaie de fusionner SystemVerilog et Verilog-AMS [Chandrasekaran 10]. Dans le même temps, les vendeurs de cosimulateurs proposent des solutions pour coupler les modèles SystemVerilog et Verilog-AMS.

2.5 SystemC et son extension AMS

SystemC est une bibliothèque spécifique C++ dédiée à la modélisation des systèmes hétérogènes logiciel/matériel numérique. Il est spécialement adapté pour analyser le partitionnement logiciel/matériel et simuler les interactions entre ces deux domaines. Les concepts de la modélisation système sont intégrés dans SystemC. SystemC a été normalisé en 2005 par l'IEEE (IEEE 1666). Cette bibliothèque permet la description et la simulation des systèmes matériels et logiciels à partir du niveau fonctionnel et jusqu'au niveau RTL en utilisant le MoC DiscreteEvent (DE). SystemC a été largement adopté dans l'industrie ces dix dernières années. Son développement et sa normalisation sont coordonnés par l'Open SystemC Initiative (l'OSCI fusionne avec Accellera pour former l'Accellera Systems Initiative à la fin de l'année 2011) [OSCI 10a]. Les domaines d'applications de SystemC continuent de s'étendre à l'heure actuelle : un meilleur support pour les systèmes embarqués (temps réel), modélisation des systèmes AMS, meilleures performances au niveau système [Ghenassia 05][OSCI 09]. Le caractère open-source et orienté-objet de SystemC, ainsi que la séparation entre la communication et le calcul permettent d'ajouter de nouveaux MoCs [Patel 05]. Pour la modélisation des systèmes numériques matériels et logiciels, différents travaux ont permis d'obtenir des MoCs portant sur les machines à états finis (FSM), les réseaux de Petri, les réseaux de Kahn, le flot de données synchrones (SDF : Synchronous Data Flow), la réaction synchrone… SystemC-H [Patel 04], HetSC [Herrera 06], UMoC++ [Mathaikutty 05] sont des exemples de ces MoCs. Leurs implémentations peuvent être très différentes. Chacun introduit de nouvelles règles quant à l'écriture d'un module par le programmeur. Certains MoCs sont implémentés en tant que canaux utilisant des évènements SystemC – donc numériques – afin de planifier leur exécution à l'aide du noyau à évènements discrets de SystemC. D'autres utilisent un noyau dédié afin d'accélérer leur exécution. Dans [Patel 08], il est montré comment ces différents outils peuvent être combinés afin d'obtenir une méthodologie de conception au niveau système pour les systèmes numériques hétérogènes matériel/logiciel.

En 2012, la version 2.3 de SystemC intègre la modélisation transactionnelle TLM (Transaction Level Modeling) dans le langage [Accellera 12]. TLM a pu être développé grâce à l'amélioration de la communication par canal virtuel : les communications sont abstraites.

SystemC connait un certain succès dans l'industrie : STMicroelectronics et NXP l'on adopté après que Synopsys et Coware aient participé au développement des premières versions. TLM permet de développer efficacement les logiciels embarqués. Différents MoCs permettent une approche top-down depuis le niveau système jusqu'au niveau des transferts entre les registres RTL. Ainsi, ce langage est particulièrement bien adapté lors de la phase d'exploration architecturale de la partie numérique.

La modélisation TLM permet de disposer du modèle d'un système en cours de développement. Le modèle TLM est disponible bien plus tôt que le modèle RTL correspondant. C'est ce qui permet aux ingénieurs logiciels de commencer le développement du logiciel embarqué en se basant sur le prototype virtuel TLM. Lorsque le modèle RTL est achevé, il peut être implanté sur FPGA afin de créer une plate-forme matérielle de prototypage. L'architecture matérielle peut alors être vérifiée, le modèle TLM, lui-même vérifié au préalable avec des méthodes formelles ou semi formelles [Ferro 11] servant de référence. Pour cette tâche de vérification, un banc de test est nécessaire, il doit être capable d'exécuter des vecteurs de tests sur le modèle RTL. SystemVerilog, un langage capable de générer des vecteurs de tests aléatoires et de vérifier les résultats donnés pour chaque vecteur est performant dans cette tâche de vérification. La vérification simultanée des parties matérielles et logicielles est possible en utilisant la cosimulation SystemC/SystemVerilog [Berman 05][Aynsley 09].

L'utilisation conjointe de SystemC et SystemVerilog permet la vérification des systèmes hétérogènes numériques incorporant des parties logicielles et matérielles. Toutefois, il ne permet pas de travailler sur l'hétérogénéité électrique (analogique/numérique). Aujourd'hui, les systèmes sur puces ou les systèmes en boitiers SoC-SiP (System on Chip – System in Package) incorporent directement des capteurs et/ou des actionneurs. Ces éléments, permettant l'interaction avec le monde extérieur, sont bien entendu multiphysiques. Les capteurs permettent de mesurer une grandeur physique et d'envoyer au système une grandeur électrique (analogique) correspondante, celle-ci étant par la suite convertie en valeur numérique puis utilisée par l'unité de calcul lors du traitement numérique. Le traitement numérique peut éventuellement déclencher un signal de commande sur un actionneur, ce dernier permettant d'influer sur le monde extérieur.

Avec la complexification des systèmes embarqués et les interactions de plus en plus importantes entre les domaines physiques, la nécessité pour les industries de l'automobile, de l'aéronautique, des télécommunications et bien sûr des semi-conducteurs, de disposer de prototypes virtuels se fait ressentir. Ces prototypes virtuels ne se limitent plus à des blocs matériels numériques : aujourd'hui les industriels souhaitent pouvoir développer des modèles de blocs analogiques ou à signaux mixtes. Les solutions basées sur la cosimulation (utilisation conjointe de divers outils, par exemple : VHDL + SPICE + C) ne sont pas assez simples d'utilisation : les outils de CAO n'ayant pas été développés pour interagir entre eux.

Etant donné le succès de SystemC dans l'industrie, l'idée de l'améliorer et de proposer des extensions s'est imposée. Aucune solution complète n'était disponible pour modéliser et simuler les systèmes hétérogènes. D'une part, des outils tels que MATLAB/Simulink et Ptolemy II permettaient de travailler au niveau système, d'autre part les langages de description matérielle permettaient les descriptions de bas niveau. Il fallait donc un outil permettant de modéliser à un niveau d'abstraction intermédiaire et qui soit capable d'incorporer les dépendances matériel/logiciel. SystemC étant déjà performant pour la création de prototypes virtuels numériques matériel/logiciel, le but était d'étendre ses capacités aux autres types d'hétérogénéités : hétérogénéités électrique et multiphysique. Durant les années 2000, un grand défi dans le domaine était la conception des systèmes communicants comme les réseaux de capteurs, c'est donc tout naturellement que les extensions de SystemC ont été élaborées sur la base de la modélisation des communications sans fil entre systèmes [Grimm 08].

Dans [Al-Junaid 05], SystemC est étendu à SystemC-A, supportant les variables et les composants analogiques, les performances de ce langage sont présentées. Des MoCs dédiés à l'analogique sont introduits. Au sein d'un même modèle, différents niveaux d'abstraction sont possibles : niveau système, niveau comportemental, niveau circuit électrique. Au niveau du circuit électrique, chaque composant d'une netlist contribue à l'initialisation d'une matrice de type Analyse Nodale Modifiée (MNA) en spécifiant les contributions à chaque terminal conservatif. En fait, un système d'Equations Différentielles Algébriques (DAE) est construit. Des interfaces spécifiques permettent de manipuler les interconnections entre les modèles analogiques et numériques. Une interaction numérique-analogique est réalisée en convertissant un signal numérique en un signal analogique en se basant sur la méthode d'Euler implicite avec un très petit pas de temps. L'interaction analogique-numérique est réalisée en détectant le passage de seuils. Le noyau de simulation de SystemC est modifié afin d'y inclure l'exécution d'un solveur analogique. Un jeu de composants élémentaires est fourni avec ce langage. Dans [Al-Junaid 06], SystemC-A est utilisé pour modéliser un système de suspension active pour siège de véhicule automobile. Le cas d'étude montre une bonne correspondance entre le modèle SytemC-A et le modèle VHDL-AMS. Cependant, la solution présentée présente l'inconvénient suivant : il faut modifier le noyau de simulation SystemC afin de coupler le solveur analogique au solveur numérique. Une autre solution aurait été de fournir une couche d'abstraction au-dessus de SystemC, donc plus élevée, afin de pouvoir intégrer de manière parallèle plusieurs MoC à temps continu. Aussi, la façon de définir les contributions à la matrice du système est très proche du niveau circuit et ne semble pas très appropriée pour les systèmes hétérogènes complexes. D'autre part, la modification du noyau SystemC entraine une non-conformité avec la norme établie.

SystemC WMS (Wave Mixed Signal) s'appuie sur la théorie des ondes et propose une méthode permettant de l'appliquer aux circuits analogiques et mixtes. Les modules communiquent entre eux par échange d'énergie au moyen d'ondes. Un module reçoit une onde incidente, le signal d'entrée, et renvoie une onde réfléchie, le signal de sortie. Comme

dans une ligne, le canal est traversé par l'onde incidente et l'onde réfléchie. Cette approche permet de résoudre les systèmes pouvant être décrits par des équations différentielles, linéaires ou non. Cette méthode étant basée sur l'échange d'énergie, elle peut être employée pour des domaines physiques autres que l'électricité. Etant donné qu'il faut résonner en termes d'ondes, des notions de radiocommunications sont nécessaires. En effet, des modules associés en parallèle devront être liés par un adaptateur d'onde et non par une dérivation comme on le fait en électricité. Les vitesses de simulation de SystemC WMS n'étant pas supérieures à celles de MATLAB/Simulink, cette méthode n'a pas eu de succès. Il était en effet impossible pour le simulateur de calculer de façon indépendante du temps simulé [Orcioni 08].

La solution apportée par SystemC AMS, normalisé par l'OSCI en 2009 présente l'avantage de pouvoir interagir très aisément avec SystemC. Cette plate-forme permet donc de modéliser sans se soucier des problèmes de cosimulation toutes les parties d'un système, qu'elles soient logicielles, matérielles numériques, analogiques, RF ou mécaniques, hydrauliques…D'autre part, cette plate-forme permet la modélisation à haut niveau d'abstraction, ce qui implique des durées de simulation réduite. Cependant, les différents MoCs permettent de construire des modèles plus ou moins abstraits : l'exploration architecturale est donc possible. Il est aussi possible d'ajouter des solveurs spécifiques, si besoin est. Les codes de SystemC et de SystemC AMS sont ouverts, cela facilite les évolutions de ces langages. SystemC est déjà bien répandu dans l'industrie et certains industriels ont déjà effectué certains travaux reposant sur SystemC AMS.

En 2002, un groupe de travail, l'OSCI AMS Working Group (AMSWG) [Accellera 02] s'est constitué afin de mener à bien le projet SystemC AMS. Dès 2003, les spécifications de l'extension AMS de SystemC ont été publiées. Au départ, SystemC AMS ne comportait que deux MoCs, celui faisant intervenir des circuits électriques : conservatif et basé sur l'analyse nodale modifiée et un autre de type flot de données qui permettait de construire des modèles analogiques de façon algorithmique. Un troisième MoC a été ajouté ensuite, il permettait de modéliser facilement les systèmes asservis linéaires en incluant notamment la gestion des boucles de rétroaction. A partir de 2006, l'AMSWG a pour mission de normaliser SystemC AMS. Certaines tâches sont considérées comme étant prioritaires, à la demande des industriels. Dans un premier temps, il faudra s'atteler aux circuits électriques linéaires. Ainsi, les solveurs analogiques ne résoudront que les systèmes d'équations linéaires, les comportements non linéaires pouvant être modélisés par des algorithmes grâce au MoC de type flot de données. Le pas de temps de simulation est fixe, pour des raisons de simplicité et d'efficacité [Vachoux 03], [Einwich 10].

SystemC AMS est interfacé avec SystemC : une couche de synchronisation permet cela. Elle garantit la séparation de SystemC et SystemC AMS. Tant que l'interface est respectée, les deux langages évoluent indépendamment. Le solveur numérique de SystemC et le solveur analogique de SystemC AMS progressent de façon séparée au cours de la simulation afin de

maintenir la vitesse et la précision de celle-ci. Aux moments cruciaux, la synchronisation est toutefois assurée. Enfin, SystemC AMS est un outil souple : le concepteur peut ajouter une extension à SystemC AMS en la connectant à la couche de synchronisation comme par exemple dans [Uhle 10] ou dans [Mähne 11]. Ces avantages sont déterminants par rapport à SystemC-A.

La journée SystemC AMS de Dresde, en 2011 [Accellera 11] a permis de montrer l'intérêt suscité par SystemC AMS, que ce soit auprès des industriels ou des chercheurs. L'Accellera Systems Initiative a donc choisi de continuer à faire évoluer SystemC AMS en fonction des demandes des utilisateurs. La version 2.0 de SystemC AMS intègre désormais les simulations à pas variable, atout crucial pour un simulateur analogique [Accellera 12].

SystemC AMS défini les formalismes essentiels pour la modélisation comportementale des systèmes AMS à différents niveaux d'abstraction. Il existe trois MoCs permettant de mettre en œuvre ces formalismes : Timed Data Flow (TDF), Linear Signal Flow (LSF) et Electrical Linear Network (ELN). Un modèle de calcul SystemC AMS est un formalisme de modélisation utilisé pour créer un modèle exécutable. De la même façon qu'en SystemC, un modèle SystemC AMS est créé par une association de modules. Une association de modules d'un même MoC est un cluster. Le MoC ELN repose sur les composants électriques linéaires. Les MoCs TDF et LSF s'appuient sur des modules et des signaux abstraits. Un seul MoC permet à l'utilisateur de définir ses propres modules, le MoC TDF. Les MoCs LSF et ELN ne permettent théoriquement pas cela : l'utilisateur ne peut que créer un module en réalisant un assemblage de primitives. Les trois paragraphes suivant présentent brièvement les trois MoCs de SystemC AMS. Ils seront détaillés plus longuement au cours du chapitre suivant.

Le MoC TDF est celui qui permet le plus de liberté à l'utilisateur qui a la possibilité de créer ses propres modules. L'utilisateur décrit le comportement de son module de façon algorithmique, comme pour des fonctions C, ce qui permet de modéliser des non-linéarités aisément. Il est possible d'utiliser des classes prédéfinies permettant d'instancier des fonctions de transfert par exemple. Il existe aussi des classes prédéfinies permettant de convertir un signal TDF vers une tension (ELN) ou un signal LSF et inversement. C'est aussi ce MoC, qui est le plus proche du monde numérique, il dispose ainsi de classes permettant de convertir un signal TDF vers un signal numérique pur (SystemC). Dans ce MoC, le temps est discrétisé et l'amplitude est continue, les signaux sont donc des signaux échantillonnés. Les modules TDF sont interconnectés au moyen de ports TDF reliés par des signaux TDF.

Le MoC LSF permet à l'utilisateur de modéliser de façon efficiente les systèmes asservis linéaires. Une bibliothèque de primitives correspondant aux fonctions que l'on retrouve habituellement dans les systèmes asservis (bloc gain, intégrateur, dérivateur…) est disponible. L'utilisateur créé un modèle LSF en réalisant un assemblage de primitives LSF. Les primitives LSF réalisent toutes des fonctions linéaires. Un cluster LSF introduit des équations qui seront résolues par un solveur d'équation différentielles algébriques. En LSF, les signaux

sont à temps continu. Les modules LSF sont interconnectés au moyen de ports LSF reliés par des signaux LSF L'utilisateur ne peut théoriquement pas créer ses propres modules LSF : un correcteur PID est ainsi réalisé par un assemblage de primitives LSF connectés par des signaux LSF mais le module encapsulant cet assemblage est de type TDF. De plus, l'utilisateur ne peut pas créer sa propre primitive LSF.

Le MoC ELN permet de modéliser un circuit électrique linéaire. Comme en LSF, l'utilisateur ne peut pas créer ses propres modules. Il n'est donc pas possible d'utiliser des composants non linéaires. Les composants électriques linéaires peuvent être instanciés entre des nœuds électriques afin de créer un circuit. Comme un cluster LSF, un cluster ELN introduit des équations qui seront résolues par un solveur d'équation différentielles algébriques. En ELN, le potentiel électrique de chaque nœud ainsi que le courant dans chaque branche doivent être calculés, ce MoC fait donc intervenir la notion de conservativité de la puissance dans ses modèles. Ainsi, un modèle ELN introduira plus d'équations qu'un modèle LSF comportant autant de primitives. Les modules ELN sont interconnectés au moyen de terminaux reliés par des nœuds ELN.

Durant une simulation SystemC AMS, une séquence d'appel des fonctions de traitement est exécutée. C'est la phase d'élaboration, prélude à toute simulation SystemC AMS qui permet de déterminer comment les fonctions de traitement sont appelées. Au cours de la phase d'élaboration, une autre étape a lieu : la vérification des connections entre les modules. Malheureusement, le produit de l'élaboration ne peut être réutilisé : avant chaque simulation, la phase d'élaboration a lieu et celle-ci est assez coûteuse en temps, notamment en ce qui concerne des modélisations ELN ou LSF.

2.6 Modélisation des systèmes hétérogènes

Dans [Bonnerud 01], les signaux analogiques ont été définis en tant que nouveaux objets SystemC, pour ce faire, une nouvelle plate-forme de simulation SystemC a été développée. Cela fonctionne toujours de façon évènementielle, mais les horloges virtuelles permettent d'optimiser la simulation des modules analogiques et mixtes. Des modèles structurels comme les convertisseurs analogique-numérique peuvent être construits à l'aide d'une bibliothèque de modèles comportementaux analogiques et mixtes. Cependant, cette approche n'a pas été généralisée et ne permet pas de modéliser n'importe quels types de comportements analogiques et l'utilisation du formalisme évènementiel limite son application aux comportements communicants au moyen de signaux et aux pas de temps fixes.

Dans [Biagetti 04], il est décrit une méthode permettant d'écrire des modèles de composants analogiques en utilisant la bibliothèque standard SystemC et son noyau de simulation. Les modules analogiques sont implémentés comme des modules SystemC normaux ayant tout de même une architecture spécifique permettant de manipuler les pas de temps adaptatifs. Les modules analogiques et mixtes sont simulés de façon évènementielle mais chacun est réactivé en utilisant son propre pas de temps. Les équations différentielles doivent être discrétisées de

façon manuelle avec un pas de temps propre. Cette approche permet aussi la modélisation des comportements communicants au moyen de signaux.

L'approche décrite dans le paragraphe précédent a été étendue dans [Vachoux 06] avec pour but de modéliser les systèmes conservatifs. Ce travail, appelé SystemC WMS permet l'implémentation de modules analogiques qui communiquent les uns avec les autres en échangeant de l'énergie au travers de canaux. Ces canaux sont des interfaces analogiques pouvant supporter différents domaines physiques. Il est possible d'interconnecter des modules décrivant des comportements de composants physiques. Il n'est pas nécessaire de modifier la bibliothèque SystemC, seul le standard de communication du noyau SystemC est utilisé. L'interface par canal simplifie l'interconnexion de modules développés indépendamment, il n'y a plus les problèmes de définition des variables « trough » et « across » lors de l'implémentation du module. Un inverseur a été modélisé comme exemple d'application en utilisant SystemC WMS et simulé en utilisant un solveur d'équations différentielles ordinaires (ODE) Adams-Bashforth du quatrième ordre. Les résultats de simulation montrent une bonne correspondance avec ceux donnés par un modèle équivalent crée en utilisant la Power toolbox de Matlab Simulink et simulé en utilisant `ode15s`. La simulation est environ cinq fois plus longue en SystemC WMS. Cela peut en partie être attribué aux différents solveurs ODE. Néanmoins, les performances de simulations de SystemC WMS sont limitées par le fait qu'à chaque pas de temps, plusieurs évènements discrets sont programmés, invoquant le noyau de simulation SystemC. Ainsi, les parties discrètes et continues ne peuvent pas fonctionner indépendamment l'une de l'autre.

Dans [Vachoux 03], SystemC AMS, extension de la plate-forme de modélisation SystemC pour la description et la simulation des systèmes analogiques et mixtes voit son contexte de développement défini. Dans [Vachoux 05], le prototype SystemC AMS développé dans ce contexte est présenté en détail. Deux formalismes, ou MoCs sont implémentés : l'un étant un MoC permettant de modéliser les circuits linéaires en fournissant une bibliothèque de primitives électriques, l'autre étant plus abstrait, de type SDF, et permettant la modélisation des comportements de type traitement du signal ainsi que des comportements de type temps continu en utilisant le sur-échantillonnage. Ces deux MoCs sont synchronisés avec le noyau de simulation numérique de SystemC grâce à une couche de synchronisation, permettant ainsi la simulation des systèmes à signaux mixtes en utilisant différents MoCs en parallèle.

Cette première version de SystemC AMS a été évaluée en fonction de ses performances dans le domaine d'application des MEMS qui sont impliqués dans plusieurs domaines physiques et soumis aux lois de conservation de l'énergie. Ce prototype a été testé sur un système de navigation inertiel intégrant des capteurs mécaniques. Dans une première approche, les pièces mécaniques tels que les ressorts, masses ou amortisseurs sont modélisés par leurs équivalents dans le domaine électrique, soit respectivement les capacités, inductances et résistances qui disposent de modèles SystemC AMS. Cette approche conduit à construire le circuit équivalent électrique de la partie mécanique, et à se retrouver, en fin de compte, avec un modèle global

électrique analogique. Dans une seconde approche [Markert 07], la partie mécanique est modélisée sous forme de schéma-bloc (non-conservatif) et simulée en utilisant le MoC SDF. Cette seconde approche permet un gain en termes de temps de simulation. Les résultats donnés par les deux modèles sont conformes à ceux donnés par un modèle de référence VHDL-AMS. Cependant, SystemC AMS n'étant pas pensé pour la modélisation des systèmes multiphysiques, le travail de modélisation s'avérait plus complexe qu'en utilisant VHDL-AMS. Cet inconvénient semblait contrebalancé par le fait que SystemC AMS était largement plus performant que VHDL-AMS lorsqu'il s'agissait de modéliser les parties numériques et logicielles permettant de traiter le signal de sortie du MEMS.

Dans [Caluwaerts 08], des conclusions similaires apparaissent. Un hacheur électromécanique est ici pris à titre d'exemple. Il est constitué d'un résonateur, d'une capacité variable, d'une pompe de charge et d'un circuit flyback. Pour décrire le système, les auteurs ont assemblés des primitives électriques et des modules qu'ils ont eux-mêmes défini en utilisant le MoC SDF. Le comportement non-linéaire de la diode est modélisé par une résistance variable, dont le signal de contrôle est fixé par un module recevant une mesure de la tension aux bornes de cette même résistance. Cette boucle de retour impose d'avoir un très faible pas de temps par rapport aux fréquences mises en jeu dans le circuit pour minimiser l'erreur, en effet un module SDF introduit obligatoirement un délai égal à un pas de temps. SystemC AMS se retrouve donc limité par l'impossibilité de modéliser les comportements non-linéaires de façon raisonnable et par son absence de pas de temps adaptatif.

Dans [Herrera 07], il est montré comment coupler le simulateur SystemC AMS 0.15RC4 [Vachoux 05] avec HetSC [Herrera 06], pour pouvoir supporter en parallèle un plus large panel de MoCs. SystemC AMS est adapté de façon à pouvoir être utilisé sur toute la gamme des spécifications entrant en ligne de compte dans les systèmes embarqués hétérogènes. Les problèmes sémantiques et syntaxiques résultant de l'utilisation conjointe des deux bibliothèques sont abordés, et particulièrement les interfaces fournies par les divers MoCs pour les interactions en cours de simulation. Ce papier fait apparaître le besoin de normaliser la sémantique de synchronisation entre les différents MoCs. Dans [Zaidi 10], il est montré comment coupler le MoC SDF de SystemC AMS avec Verilog Procedural Interface (VPI) au moyen d'un simulateur AMS dans le but de réaliser des cosimulations de systèmes analogiques et mixtes à plusieurs niveaux d'abstraction. Une grande partie de ces systèmes est modélisée à haut niveau avec SystemC AMS et certains blocs sont remplacés par des modèles comportementaux plus détaillés tels que des modèles VHDL-AMS, Verilog-AMS ou même par des modèles se situant au niveau circuit, comme des modèles SPICE. Ces deux travaux démontrent les avantages de l'architecture SystemC AMS, qui facilite l'intégration de formalismes et d'outils très différents et fournit une plate-forme de simulation efficiente permettant de réaliser un travail global sur les systèmes hétérogènes, des spécifications à la vérification, en passant par la modélisation et la conception.

Le développement du prototype SystemC AMS a été encadré par le groupe de travail SystemC AMS dans le but de généraliser et de normaliser les concepts introduits par SystemC AMS. Ce groupe de travail, accompagné de sociétés de l'industrie des semi-conducteurs a intégré le consortium OSCI qui coordonne les travaux en relation avec SystemC. Depuis sa création en 2006, le groupe de travail OSCI AMS a eu pour but de promouvoir les extensions à l'analogique et aux signaux mixtes de SystemC auprès des industriels. Le groupe de travail, en se basant sur la spécification des besoins [Vachoux 03] a publié le manuel de référence (LRM), devenu une norme OSCI en mars 2010 [OSCI 10b]. Un guide de l'utilisateur a aussi été publié au même moment [Barnasconi 10]. Dans cette première version, les extensions AMS de SystemC répondent aux besoins de modélisation des comportements continus en fournissant trois MoCs, chacun permettant une description à un niveau d'abstraction spécifique : TDF, LSF et ELN. Ces MoCs sont bien appropriés à la modélisation des hétérogénéités quelles qu'elles soient.

Des essais avaient été réalisés avec le prototype de SystemC AMS pour l'étendre aux modèles conservatifs à équations algébriques dynamiques non-linéaires. Dans [Einwich 06], les auteurs présentent l'exemple multiphysique d'un micro-relai. Le MoC non-linéaire proposé définit une nouvelle classe de modules qui autorise la description de leur comportement énergétique. Pour réaliser cela, chaque module définit sa contribution aux valeurs « through » ou courants des nœuds auxquels sont connectés les ports du module dépendant des valeurs « across » ou tensions et de leurs dérivées. Des équations supplémentaires doivent être données sous la forme : *0 = F(t, vports, v'ports, varsports, vars'ports)*. La syntaxe proposée ressemble à celle de Verilog-AMS mais reste du C++. L'avantage de cette méthode réside dans le fait que ce MoC s'intègre parfaitement dans SystemC AMS sans qu'aucune modification ne soit nécessaire. En effet, la couche de synchronisation de SystemC AMS est utilisée afin d'interfacer ce MoC avec le MoC SDF de SystemC AMS et le MoC DE de SystemC. Ainsi, des systèmes hétérogènes complets peuvent être simulés au moyen de modèles décrits grâce à plusieurs MoCs. Dans [Uhle 10], il est expliqué comment l'approche précédente a été généralisée et raffinée. Les erreurs dans la construction des modèles, comme des ports électriques non connectés par exemple sont désormais signalées dès la compilation. La syntaxe de description de modèles comportementaux conservatifs a été améliorée afin d'être plus agréable pour l'utilisateur. Les nouvelles constructions de langage ont été calquées sur la norme SystemC AMS de l'OSCI. Un solveur non-linéaire peut générer des évènements basés sur le franchissement de seuils ainsi que revenir en arrière pour réagir à des évènements. Les capacités de ce MoC sont démontrées sur un exemple électromécanique complexe, un support de levage de vitres. Des parties mécaniques, magnétiques et électriques sont donc couplées. Les deux approches [Einwich 06][Uhle 10] ne permettent pas d'obtenir un véritable avantage sur VHDL-AMS en termes de durée de simulation. Cependant, sur d'autres modèles où les parties numériques logicielles et matérielles sont prépondérantes par rapport aux parties analogiques, des modèles abstraits comme ceux de SystemC AMS ont nettement l'avantage sur ce point.

Dans [Hartmann 09], le modèle SystemC AMS d'un système de contrôle est donné. C'est un contrôle embarqué d'une grue, ce modèle ayant déjà été utilisé dans [Moser 99]. Un solveur externe Runge-Kutta 4 est intégré à la simulation, remplaçant le solveur de représentation d'état de SystemC AMS. Cette approche est intéressante, le solveur externe étant encapsulé de telle façon qu'il ait la même interface que le solveur de représentation d'état, il y a très peu de modifications à opérer sur le modèle.

Dans [Damm 08a] et [Damm 08b], les auteurs montrent comment on peut construire le modèle transactionnel d'un système AMS. Le modèle est de haut niveau d'abstraction, réalisé avec les MoCs les plus abstraits. La partie numérique du système (un système de communication) est modélisée en SystemC TLM LT (Loosely Timed), la partie analogique est modélisée avec le MoC TDF de SystemC AMS. Le découplage temporel entre les différents blocs, analogiques ou numériques permet d'utiliser les deux MoCs efficacement. Le décalage temporel entre les blocs doit néanmoins être contenu afin d'éviter des erreurs dans l'ordre d'arrivée des paquets. La modélisation transactionnelle des parties analogiques avec SystemC AMS permet un gain important en termes de vitesse de simulation [Lévêque 10] [Massouri 10] [Cenni 12].

D'autres travaux ont montré, sur des exemples concrets les capacités de SysemC AMS. Dans [Alassir 06] [Alassir 07], les auteurs présentent un contrôleur de bus I2C, le modèle comporte une partie analogique et une partie numérique. Il faut signaler un nombre important de travaux concernant les réseaux de capteurs, parmi lesquels [Vasilevski 07a], [Vasilevski 07b], [Vasilevski 08a], [Vasilevski 08b] et [Hörmann 11]. Le développement de prototypes virtuels de capteurs d'image a aussi été abordé dans [Cenni 11a], [Cenni 11b], [Cenni 11c]. Ce même auteur a aussi travaillé sur un capteur chimique à ondes acoustiques de surface [Cenni 09]. Le site internet du groupe de travail SystemC AMS donne accès à ces travaux.

Enfin, dans [Mähne 11], il est présenté un MoC SystemC AMS basé sur les bond graphs. Le bond graph est un outil universel permettant d'abstraire les domaines physiques tout en maintenant le concept de modèle conservatif : on ne se concentre plus que sur les interactions. L'auteur a développé un ensemble de classes permettant d'assembler un modèle bond graph. L'extension SCAX permet de vérifier la cohérence des unités de mesure. Ainsi, un bloc qui délivre une vitesse exprimée en kilomètres par heure ne peut pas être connecté à un bloc ayant comme entrée une vitesse exprimée en miles par heure.

Venons-en maintenant aux applications dans l'industrie de SystemC AMS et des méthodologies associées. Dans [Cenni 12], l'auteur expose ses travaux portant sur la modélisation d'un capteur d'image CMOS. Un modèle conservatif basé sur le MoC ELN a d'abord été créé, chaque pixel du capteur étant modélisé par trois photodiodes. Un modèle de plus haut niveau a été créé en ne considérant qu'un seul pixel qui balaye la zone image. Ce modèle est non conservatif et construit à partir du MoC TDF. Il est soixante-dix mille fois plus rapide que le modèle ELN et deux millions et demi fois plus rapide qu'un modèle

VHDL-AMS. Ce gain de temps a un impact énorme sur le flot de conception global du système embarqué. En effet, les équipes de développement logiciel auraient normalement débuté leur travail une fois le prototype matériel disponible, mais dans ce cas, il leur a été fourni un prototype virtuel du capteur d'image neuf mois avant sa réception. Ceci entraine une plus grande fiabilité du logiciel embarqué ainsi qu'un gain non négligeable en termes de time-to-market.

Dans [Pêcheux 05], les auteurs s'appuient sur l'exemple d'un modèle d'airbag afin de démontrer comment VHDL-AMS et Verilog-AMS peuvent être utilisés pour concevoir des modèles comportementaux. Ceux-ci ont la précision nécessaire pour pouvoir valider les performances du système, tout en maintenant des durées de simulation raisonnables. La réaction chimique permettant de gonfler l'airbag est aussi modélisée.

Dans [Frevert 05], des méthodes de modélisation et de simulation pour la conception de systèmes RF sont présentées. Différents composants RF sont décrits à haut niveau d'abstraction en VHDL-AMS ou en Verilog-A. Un flot de conception AMS convenant aux systèmes RF est présenté.

Dans [Mähne 06], l'auteur présente le prototypage virtuel de transducteurs MEMS. La création de modèles comportementaux est automatisée par une méthode de réduction d'ordre.

2.7 Positionnement de nos contributions par rapport à l'état de l'art

Nous avons vu que de nombreux cas d'étude ont été étudiés en utilisant SystemC AMS. Les modèles créés ont souvent été comparés avec des références telles que Matlab Simulink, VHDL-AMS ou Verilog-AMS. Ces comparaisons ont porté sur la précision ainsi que sur la durée de simulation. En revanche, aucune comparaison n'a été effectuée entre les différents MoC de SystemC AMS. Une première partie de notre travail portera sur cette analyse.

D'autre part, nous pensons qu'un des points fort de SystemC AMS : le fait que ce soit un environnement de modélisation unifié pour le logiciel, le matériel analogique et numérique ainsi que pour d'autres domaines physiques comme la mécanique, n'a pas été réellement exploité. En effet, nous pensons qu'un tel environnement se prête bien au développement de méthodes automatiques d'aide à la conception et nous tenterons de le démontrer en présentant une méthode de raffinement, une méthode d'abstraction de modèles ainsi qu'une méthode permettant d'enrichir les modèles de haut niveau avec la consommation. Ces développements portant essentiellement sur la partie analogique ont été réalisés sous SystemC AMS.

Enfin, nous présenterons nos travaux qui portent sur la modélisation des systèmes multiphysiques par circuits équivalents ou par bond graphs sous SystemC AMS. Ces travaux vont de la génération automatique de modèles bond graphs à partir de données de bas niveau jusqu'à l'extraction du modèle mathématique à partir d'un bond graph.

3. Niveaux d'abstraction, Modèles de Calcul et styles de modélisations

L'évolution de la complexité des circuits intégrés, a induit l'essor et le développement des outils de conception assistée par ordinateur : simulateurs analogiques, numériques, RF, éléments finis… Ainsi il est possible de modéliser puis de simuler toutes les composantes d'un système. Cependant, la validation globale du système hétérogène requiert la simulation simultanée de toutes les composantes interconnectées. Cette simulation est indispensable dans le flot de conception des systèmes complexes. Ce point demeure problématique. En effet, les outils disponibles n'ont pas été conçus dans un but de cosimulation. Les interconnexions peuvent donc être difficiles à mettre en œuvre. Même en cas de réussite sur ce point, les performances de simulation s'avèreront souvent limitées car la synchronisation de tous les outils de simulation peut demander beaucoup de ressources. Enfin, bien souvent, les outils nécessaires ne sont pas libres, et les licences requises peuvent être très onéreuses.

L'extension de SystemC à la modélisation des modules analogiques et à signaux mixtes SystemC AMS se propose de répondre à ces problèmes. Différents niveaux d'abstraction et différents modèles de calcul permettent de modéliser l'ensemble d'un système. L'utilisation du couple SystemC/SystemC AMS permet de ne plus utiliser qu'un seul outil, basé sur le langage C++, pour modéliser toutes les composantes d'un système. Les principaux problèmes sont désormais les suivants :

- quel style de modélisation utiliser pour telle ou telle partie/fonction du système ?
- comment interconnecter les différents modèles de calculs ?

Le niveau d'abstraction choisi permet de déterminer plus aisément le modèle de calcul et le style de modélisation à utiliser. Toutefois, modèles de calcul et niveaux d'abstraction ne sont pas liés : nous ne pouvons pas dire avec certitude que tel modèle de calcul (et lui seul) correspond à un niveau d'abstraction. Un niveau d'abstraction peut être modélisé de différentes façons et un modèle de calcul peut être utilisé à différents niveaux d'abstraction [Paugnat 12]. Le choix du modèle de calcul sera bien souvent un compromis entre performance et durée de simulation. Par exemple, le modèle ELN, le seul qui permette une modélisation conservative ne devra pas être le seul modèle de calcul utilisé dans la modélisation d'un système complet. Tout d'abord, parce qu'il se limite aux composants linéaires : il atteindra rapidement ses limites. Toutefois, des travaux ont été menés afin de développer une extension permettant la modélisation des composants non-linéaires [Uhle 10]. D'autre part, le temps de calcul demandé peut s'avérer très grand. Nous montrerons par la suite que le temps de calcul d'un modèle ELN augmente de façon exponentielle avec le nombre de composants. Par ailleurs, un modèle doit être le plus « homogène » possible. Nous entendons par là que les différentes parties du système doivent être modélisées à des niveaux d'abstraction équivalents. L'idée est de se baser sur le travail présenté dans [Paugnat 11], à

savoir que l'on peut raffiner les parties unes à unes afin de maintenir des durées de simulation raisonnables. Le travail cité précédemment propose de revenir au modèle d'abstraction *n* après avoir raffiné et trouvé le modèle d'abstraction *n-1*. L'approche que nous proposerons au cours du chapitre 4 montre comment utiliser les résultats fournis par le modèle d'abstraction *n-1* afin d'enrichir le modèle d'abstraction *n*.

Ce chapitre introduit la notion de niveaux d'abstraction et les situe dans le cadre des systèmes embarqués. Il présente ensuite les différents modèles de calcul de SystemC AMS. Il ne comporte pas de comparaison avec d'autres outils de modélisation tels que VHDL-AMS ou Verilog. Le but est d'évaluer les différents modèles de calcul et de présenter lesquels sont les plus adaptés à un niveau d'abstraction donné. Nous introduisons un style de modélisation original permettant d'allier les avantages de deux modèles de calcul prédéfinis. Deux cas seront étudiés : le premier permet d'évaluer les performances des styles de modélisation sur un passage à l'échelle d'un cas simple. Le second se propose d'analyser un cas plus complexe.

3.1 Modèles de Calcul (MoCs) et styles de modélisation

SystemC permet le raffinement des systèmes numériques jusqu'au niveau cycle accurate grâce à sa plate-forme de simulation. TLM (Transaction Level Modeling), construit sur cette plate-forme et destiné à la modélisation de la communication et de la synchronisation permet de modéliser, simuler et concevoir des architectures numériques. Toutefois, le noyau SystemC n'a pas été conçu pour la modélisation et la simulation des systèmes analogiques à temps continu et il manque d'outils qui permettraient de couvrir la modélisation de ces systèmes du niveau fonctionnel jusqu'au niveau de l'implémentation. L'extension SystemC AMS est construite sur le standard du langage SystemC (norme IEEE 1666-2005). Elle définit des constructions de langage qui introduisent de nouvelles sémantiques d'exécution ainsi que des méthodologies pour la modélisation et la vérification des systèmes mixtes à haut niveau d'abstraction. Les différentes classes fournies par SystemC AMS constituent une bibliothèque compatible avec le standard SystemC.

Dans les modélisations à temps discret, les signaux ou les quantités physiques sont considérés comme des séquences de valeurs définies à des points temporels. Leurs valeurs peuvent êtres réelles ou discrètes. Bien que ces valeurs soient souvent considérées constantes entre les points d'échantillonnage, elles ne sont pas formellement définies. Les comportements sont modélisés par des procédures utilisant des signaux échantillonnés. La description de modèles comportementaux non linéaires est rendue possible, notamment grâce à l'utilisation de fonctions polynômiales. Ce type de modélisation s'applique pour les comportements décrits par des processus, mais aussi pour les comportements à temps continu lorsque le passage au temps discret n'entraîne que de faibles écarts.

La modélisation à temps continu est plus proche du monde physique : les signaux et quantités sont considérés tels que les valeurs réelles évoluent en fonction du temps. Le temps est ici une

grandeur continue. Les comportements sont décrits au moyen d'équations mathématiques pouvant inclure des dérivées. Les équations sont résolues par un solveur dédié (solveur linéaire). La modélisation à temps continu est particulièrement bien adaptée pour la description comportementale d'éléments physiques.

Sous SystemC AMS, les MoCs ELN et LSF permettent la modélisation à temps continu, les équations correspondant aux modèles sont résolues par le solveur linéaire. Le MoC TDF permet les modélisations à temps discret. Des convertisseurs de signaux permettent d'ailleurs de passer d'un signal purement numérique SystemC à un signal de type TDF. C'est donc ce MoC qui permet l'interfaçage de SystemC avec SystemC AMS.

Les modèles à temps continu peuvent être séparés en deux classes distinctes : les modèles non conservatifs et les modèles conservatifs. Un modèle non conservatif exprime un comportement en tant que flux dirigé d'une quantité exprimée en fonction du temps continu, quantité sur laquelle des fonctions telles que le filtrage ou l'intégration peuvent être appliquées. Les effets dynamiques y sont décrits, mais les interactions énergétiques entre les blocs ne sont pas prises en compte.

Les modèles conservatifs sont plus détaillés. La conservation de l'énergie doit y être respectée. Ainsi, le jeu d'équations à résoudre est plus grand et complexe que le jeu d'équations introduit par un modèle non conservatif.

Sous SystemC AMS, seul le MoC ELN permet d'écrire des modèles conservatifs de manière simple. Ce MoC est basé sur les lois de l'électricité, ainsi lors de la résolution du système d'équations par le solveur linéaire, le potentiel électrique de chaque nœud ainsi que l'intensité du courant dans chaque branche sont calculés. La conservativité de l'énergie peut donc être respectée.

3.1.1 Timed Data Flow – TDF

Le modèle de calcul TDF est basé sur le formalisme SDF (Synchronous Data Flow) de la précédente version de SystemC AMS. TDF est un style de modélisation à temps discret qui considère les données comme des signaux échantillonnés. La Figure 5 montre le principe de la modélisation TDF. Sur cette figure, les trois modules *A*, *B* et *C* communiquent. Un modèle TDF est composé d'un jeu de modules TDF connectés qui forment un cluster TDF. Un module TDF peut posséder plusieurs ports TDF, en entrée aussi bien qu'en sortie. Un module n'ayant que des ports de sortie, comme le module *A* est appelé un module émetteur. Un module n'ayant que des ports d'entrée comme le module *C* est appelé un module récepteur. Des signaux TDF sont utilisés afin de connecter les modules TDF entre eux. Chaque module TDF contient une ou plusieurs méthodes C++ (définies par l'utilisateur) calculant une fonction mathématique. Ainsi, les non linéarités peuvent être modélisées. Le comportement du cluster est défini par une composition de ses fonctions. Une fonction est traitée si les échantillons disponibles sur les ports d'entrée sont suffisants. Dans ce cas, le module TDF lit les échantillons d'entrée, la fonction utilise les valeurs lues afin de calculer les résultats qui

sont écrits sur les ports de sortie appropriés. Le pas de temps représente l'intervalle de temps entre deux échantillons. Chaque instanciation d'un module conduit à l'instanciation d'un solveur. Dans l'exemple, trois modules étant instanciés, il y aura trois solveurs instanciés. Chaque solveur résolvant les équations d'un seul module. Ces résolutions sont menées en parallèle.

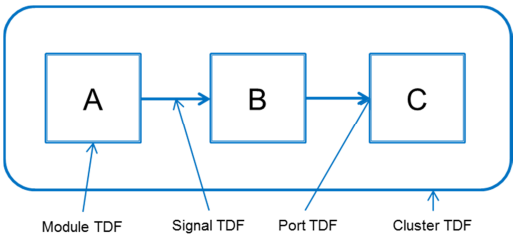


Figure 5. Modélisation TDF

3.1.2 Linear Signal Flow – LSF

Le modèle de calcul LSF permet de modéliser les comportements AMS définis en tant que relations entre variables d'un jeu d'équations linéaires algébriques. La modélisation LSF est une modélisation à temps continu utilisant des signaux réels dirigés permettant de décrire des systèmes non conservatifs. Un signal est représenté par une quantité de valeur réelle. Les modèles LSF peuvent être décrits par des schémas blocs. La Figure 6 montre un exemple de modèle LSF. Un système d'équation LSF est une association de modules élémentaires LSF connectés par des signaux LSF. La bibliothèque SystemC AMS fournit un jeu de modules LSF prédéfinis, l'utilisateur ne peut en théorie pas créer ses propres modules. Parmi les modules prédéfinis, on trouve l'additionneur, le multiplicateur, le dérivateur, l'intégrateur, etc. Le système d'équations d'un modèle LSF est calculé à partir des équations mathématiques de chacun des modules composant ce modèle. Contrairement à TDF, un seul solveur linéaire est instancié pour un cluster LSF.

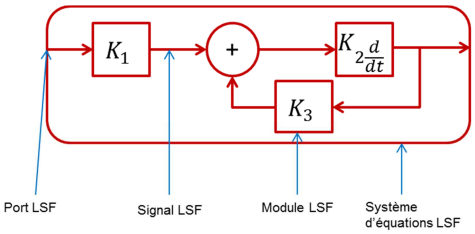


Figure 6. Modélisation LSF

3.1.3 Electrical Linear Network – ELN

Le modèle de calcul ELN permet de décrire les systèmes à temps continu et conservatifs basés sur les lois de l'électricité. Les lois de Kirchhoff sont utilisées afin de calculer le potentiel électrique de chaque nœud ainsi que le courant électrique dans chaque branche. Le réseau électrique est représenté par un jeu d'équations différentielles algébriques qui est résolu durant la simulation. Un modèle ELN est construit en instanciant des modules ELN reliés à des nœuds ELN afin former un système d'équations ELN. Le nœud de référence a toujours un potentiel électrique nul. Ici, les interfaces d'un module ELN ne pas sont pas appelées ports mais terminaux. La Figure 7 montre un exemple de modèle ELN. De même qu'en LSF, l'utilisateur ne peut pas créer ses propres modules ELN. Les primitives fournies comprennent les composants linéaires (résistance, capacité, inductance) ainsi que les différentes sources. Comme pour LSF, le système d'équations généré est résolu par le solveur linéaire. Le Tableau 2 présente un résumé des MoCs de SystemC AMS.

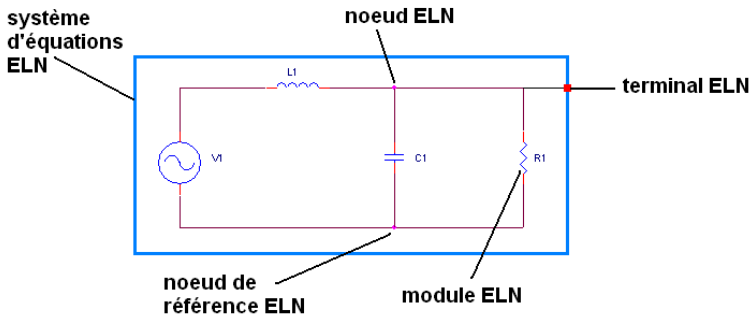


Figure 7. Modélisation ELN

Tableau 2. Résumé des MoCs de SystemC AMS

Modèle de calcul	**TDF** **Timed Data Flow**	**LSF** **Linear Signal Flow**	**ELN** **Electrical Linear Network**
Représentation temporelle	Temps discret	Temps continu	Temps continu
Interconnexions des modules	Signaux abstraits échantillonnés	Signaux abstraits continus	Signaux électriques
Description comportementale	Illimitée Modélisation des non linéarités	Opérations linéaires sur les signaux	Comportements linéaires uniquement
Traitement **Résolution**	Algorithmique	Système d'équations différentielles algébriques	Système d'équations différentielles algébriques

3.1.4 Une contribution originale, « LSF encapsulé »

Une idée originale est née au sein du laboratoire TIMA, en collaboration avec Franck Paugnat, doctorant travaillant sur le flot de conception analogique et SystemC AMS : celle de développer un style de modélisation permettant de combiner les avantages de LSF et de TDF. Nous avons donc proposé d'encapsuler les primitives LSF avec des convertisseurs TDF vers LSF en entrée et des convertisseurs LSF vers TDF en sortie. De cette façon, chaque module de base (primitive LSF + convertisseur(s) d'entrée + convertisseur(s) de sortie) instancié génèrera une instance du solveur. On dispose ainsi des primitives LSF permettant de modéliser de manière efficace les systèmes dynamiques linéaires d'une part et d'une plus grande rapidité de calcul d'autre part, puisqu'un plus grand nombre de solveurs sont instanciés. Nous n'avons plus un seul solveur devant résoudre un nombre d'équations potentiellement important mais plusieurs solveurs n'ayant qu'un nombre réduit d'équations à résoudre.

3.1.5 La bibliothèque LSF encapsulé

L'inconvénient de l'approche LSF encapsulé est que l'utilisateur pourra trouver assez rébarbatif le fait de ne plus avoir qu'un seul module à instancier mais plusieurs. Il faut aussi ajouter à ce point que le nombre de signaux internes du modèle construit va grandement augmenter. Une solution pour pallier cet inconvénient a consisté à développer un fichier d'en tête (.h) comportant tous les modules de base LSF encapsulé. Les modules de base LSF encapsulé sont en fait les primitives LSF disposant d'entrées/sorties de type TDF. Par la suite, nous établirons un comparatif de LSF encapsulé par rapport à LSF.

3.1.6 Dynamic Timed Data Flow (DTDF)

La version 2.0 de SystemC AMS, disponible depuis le printemps 2013 offre la possibilité d'utiliser un nouveau MoC : Dynamic Timed Data Flow. L'énorme avantage de ce MoC par rapport au TDF classique est qu'il permet de disposer d'un pas de simulation variable. Le pas initial est fixé par l'utilisateur. Ensuite, en cours de simulation, la valeur de l'échantillon *n* est comparée à celle de l'échantillon *n-1*. Si la variation est faible, le pas de simulation est augmenté. Par contre, si la variation est élevée alors le simulateur revient en arrière et recalcule avec un pas plus faible. Nous n'avons pas encore testé cette dernière version de SystemC AMS, mais il est évident que cette nouvelle fonctionnalité permettra d'analyser des systèmes bien plus complexes que par le passé.

3.2 Styles de modélisation et vitesse de simulation

Puisque plusieurs niveaux d'abstraction peuvent être modélisés au moyen des différents MoCs, l'utilisateur devra faire un choix. Afin d'assister l'utilisateur dans ce choix et d'obtenir le meilleur compromis vitesse de simulation/précision, nous nous proposons d'évaluer les temps de simulation des trois MoCs au moyen de deux analyses. La première analyse consiste à instancier et associer un grand nombre de fois un module très simple et d'observer le temps nécessaire à la simulation, c'est une forme de passage à l'échelle. La seconde analyse

concerne la complexité : un bloc unique est instancié mais son degré de complexité est plus élevé. Les deux cas d'étude ont été créés dans le seul but d'analyser la vitesse de simulation en fonction du style de modélisation choisi.

3.2.1 Passage à l'échelle

L'idée est ici de construire un filtre d'ordre *n* de trois façons différentes, en se basant sur les possibilités offertes par les trois MoCs. Le bloc élémentaire est un filtre passe bas du premier ordre. Le filtre global d'ordre *n* est constitué par la mise en cascade *n* fois du filtre élémentaire, c'est donc un filtre passe bas d'ordre *n*. Le signal en entrée du filtre est une tension sinusoïdale générée au moyen de la classe ELN `sca_eln::sca_vsource`.

3.2.1.1 Modèle TDF

Le MoC TDF nous offre la possibilité d'instancier des fonctions de transfert grâce à la classe `sca_tdf::sca_ltf_nd` notamment. L'utilisateur doit spécifier deux paramètres : le numérateur et le dénominateur de la fonction de transfert. Le bloc élémentaire est donc une fonction de transfert passe bas du premier ordre. Si l'on construit un filtre d'ordre *n* grâce à cette primitive en l'instanciant *n* fois, cela engendrera *n* instances du solveur. Chaque instance du solveur n'a qu'une seule équation à résoudre. Le signal d'entrée est de type ELN, il faut donc convertir ce signal en signal de type TDF pour l'interfacer avec le filtre. La source de tension engendre une équation, de même que le convertisseur. Un solveur linéaire est instancié pour calculer ces deux équations. Le nombre total d'équations à résoudre pour ce modèle est égal à *n+2*. Le nombre total de solveur est égal à *n+1*. La Figure 8 présente le modèle du filtre modélisé en TDF.

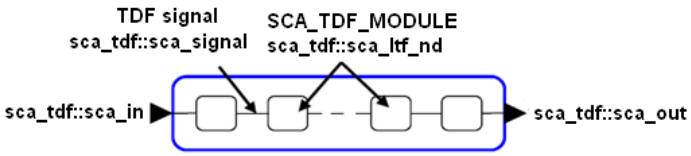


Figure 8. Modèle de filtre basé sur le MoC TDF

3.2.1.2 Modèle LSF

De même que le MoC TDF, le MoC LSF dispose de primitives permettant la modélisation sous forme de fonctions de transfert. L'équivalent LSF de `sca_tdf::sca_ltf_nd` est naturellement `sca_lsf::sca_ltf_nd`. La construction du filtre global d'ordre *n* est réalisée de la même façon qu'en TDF, cependant les *n* instances du filtre élémentaire n'introduiront qu'une unique instance du solveur. Le MoC LSF étant basé par l'intermédiaire du solveur linéaire sur l'analyse nodale modifiée, chaque nœud (qu'il faut entendre dans ce cas comme signal LSF) induit une équation supplémentaire. Ces tests ont été réalisés sous la version 1.0 bêta 2 de SystemC AMS. Cette version ne dispose pas de convertisseur ELN vers LSF et LSF vers ELN, nous avons dû passer par un signal intermédiaire de type TDF. Ainsi, la tension sinusoïdale est convertie en signal TDF puis en signal LSF. En sortie du filtre, par

souci de cohérence, le signal LSF est converti en TDF. Les convertisseurs TDF vers LSF et LSF vers TDF n'induisent aucune équation : la relation est directe, il n'y a pas de calcul engendré par le convertisseur. Les *n* instances de la primitive `sca_lsf::sca_ltf_nd` introduisent *n* équations, les *n+1* signaux LSF introduisent *n+1* équations, la source de tension et le convertisseur ELN vers TDF introduisent 2 équations. Le nombre total d'équation est égal à *2n+1*. Le nombre total de solveurs est égal à 2. La Figure 9 présente le filtre modélisé en LSF.

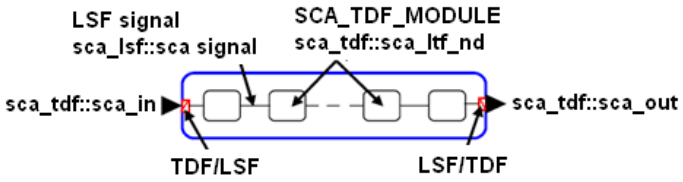


Figure 9. Modèle de filtre basé sur le MoC LSF

3.2.1.3 Modèle ELN

En ELN, le filtre élémentaire du premier ordre est modélisé à l'aide d'un circuit RC associé à un montage suiveur permettant l'adaptation d'impédance. Les primitives `sca_eln::sca_r` et `sca_eln::sca_c` modélisent respectivement la résistance et la capacité. L'amplificateur opérationnel fonctionnant en régime linéaire et considéré comme idéal est modélisé par la primitive `sca_eln::sca_nullor`. Le nœud de référence est modélisé par la primitive `sca_eln::sca_node_ref` alors que les autres nœuds du circuit sont définis par la primitive `sca_eln::sca_node`. Le modèle est entièrement construit à partir de primitives ELN, il n'apparaît donc aucun convertisseur. Le MoC ELN étant basé sur l'analyse nodale modifiée, le potentiel de chaque nœud doit être calculé, ce qui introduit une équation par nœud. Le comportement d'un condensateur est régit par une équation différentielle, donc l'ajout d'un condensateur ajoute une équation à résoudre. De même le nullor ajoute une équation au système. Une résistance dont la valeur est suffisamment grande pour ne pas être considérée comme un court-circuit est traitée comme une conductance placée en parallèle entre les nœuds auxquels elle est connectée, aucune équation n'est donc ajoutée au système lorsqu'on instancie une résistance. Ainsi, pour chaque filtre élémentaire, 3 équations sont introduites (nullor + capacité + nœud interne), ce qui correspond pour un filtre d'ordre *n* à *3n* équations. Les *n+1* nœuds du circuit introduisent *n+1* équations. Enfin, la source de tension introduit une équation, portant le nombre total d'équations du modèle à *4n+2*. Le solveur n'est instancié qu'une seule fois. La Figure 10 présente la modélisation ELN du filtre.

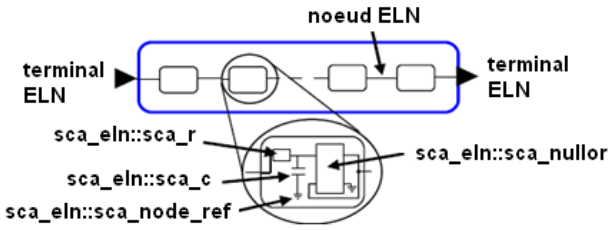


Figure 10. Modèle de filtre basé sur le MoC ELN

3.2.1.4 Modèle LSF encapsulé

En LSF encapsulé, le filtre élémentaire est construit à partir de la primitive `sca_lsf::sca_ltf_nd` connectée en amont à un convertisseur TDF vers LSF et en aval à un convertisseur LSF vers TDF. Les *n* instances de la primitive introduisent *n* équations. Chaque signal LSF introduit une équation, ainsi que chaque instance de la classe `sca_lsf::sca_ltf_nd`. Les convertisseurs n'introduisent aucune équation. Ainsi, chaque bloc élémentaire introduit un solveur ayant 3 équations à résoudre. Le filtre est toujours excité par une source de tension sinusoïdale modélisée en ELN. Un filtre d'ordre *n* introduit donc *3n+2* équations et *n+1* solveurs. La Figure 11 présente le modèle du filtre LSF encapsulé.

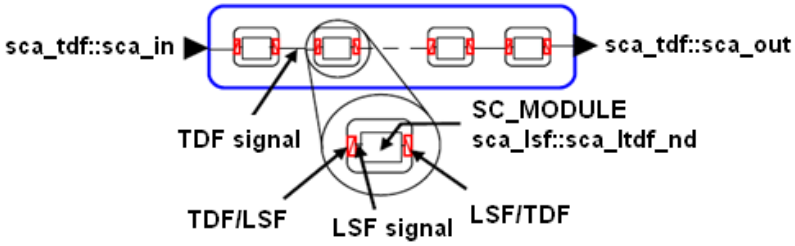


Figure 11. Modèle de filtre basé sur le style LSF encapsulé

3.2.1.5 Passage à l'échelle – Résultats

Le Tableau 3 présente tout d'abord un récapitulatif du nombre d'équations et du nombre de solveurs instanciés pour les différents modèles. Nous pouvons prévoir à partir de ces données que les modèles ELN et LSF devraient être plus long à simuler que les modèles TDF et LSF encapsulé. En effets, les 2 solveurs instanciés en LSF et l'unique solveur instancié en ELN auront un grand nombre d'équation à résoudre.

Tableau 3. Récapitulatif des styles de modélisation utilisés – Passage à l'échelle

MoC	Nombre total d'équations générées	Nombre total de solveurs instanciés
TDF	n+2	n+1
LSF	2n+3	2
ELN	4n+2	1
LSF encapsulé	3n+2	n+1

Les paramètres de la simulation sont présentés par le Tableau 4. Les *n* instances du filtre élémentaire ont toute la même fréquence de coupure. La fréquence de la source de tension sinusoïdale est fixée à cette fréquence de coupure. Le pas de temps a été choisi afin d'obtenir 1000 points par période du signal d'entée. La résolution temporelle du simulateur est choisie égale au pas de temps par souci d'optimisation.

Tableau 4. Paramètres de la simulation – Passage à l'échelle

Fréquence de coupure du filtre élémentaire	100 Hz
Fréquence du signal d'entrée	100 Hz
Durée simulée	100 ms
Pas de temps du simulateur	100 µs
Résolution temporelle du simulateur	100 µs

Les modèles ont été simulés sur un ordinateur PC, sous Linux avec les caractéristiques suivantes : CPU Intel Xeon X5570 (2.93 Ghz, 8 Mo), RAM 24 Go (1066 MHz), HDD 10000 tours/min. La version 1.0 bêta 1 de SystemC AMS a été utilisée pour évaluer les performances des MoCs. Quant à SystemC, c'était la version 2.2. Les résultats ont été comparés à Matlab Simulink, version 7.10R2010a. Le modèle Simulink était une transposition du modèle TDF vers Matlab. Le solveur Simulink utilisé est `ode1`, basé sur la méthode d'Euler, méthode employée par SystemC AMS. Les durées présentées par la suite ont été acquises au moyen de la fonction C : `clock()`. Ce sont des durées moyennes calculées à partir de 10 tests réalisés.

La Figure 12 montre clairement que le temps d'élaboration et de simulation des modèles ELN et LSF est bien plus important que celui des modèles TDF et LSF encapsulé. Ceci s'explique

par le nombre d'équations à résoudre par solveur. Les modèles ELN et LSF introduisent de plus en plus d'équations lorsque le nombre d'instances augmente alors que le nombre de solveurs reste constant est très faible (1 pour ELN, 2 pour LSF). Les modèles TDF et LSF encapsulés introduisent quant à eux un nouveau solveur par instance. Les solveurs instanciés ont alors peu d'équations à traiter et leur nombre n'augmente pas avec le nombre d'instances.

L'écart que l'on remarque entre les modèles ELN et LSF s'explique par le fait que le solveur doit résoudre près de deux fois plus d'équations dans le cas du modèle ELN. En effet, le modèle ELN est conservatif et basé sur les lois de l'électricité, le courant de chacune des branches ainsi que le potentiel de chacun des nœuds, soit deux quantités doivent être résolues. Le modèle LSF est non conservatif, un seul signal est traité.

L'écart que l'on remarque entre les modèles TDF et LSF encapsulé est dû au fait que chaque solveur instancié par le modèle du filtre introduit trois équations en LSF encapsulé et une seule en TDF. Les deux équations supplémentaires sont liées aux convertisseurs amont (TDF vers LSF) et aval (LSF vers TDF) des primitives LSF.

Enfin, la Figure 13 (un agrandissement de la Figure 12 pour un nombre d'instances inférieur à 200), montre que les différents modèles construits à partir de SystemC AMS sont plus performants que Simulink pour un nombre d'instances inférieur à 100. Au-delà d'une centaine d'instances, le modèle ELN devient plus lent que Simulink. Le modèle LSF devient lui plus lent que Simulink à partir de 200 instances. Pour un grand nombre d'instances on remarque que les modèles TDF et LSF encapsulé restent équivalents à Simulink en termes de durée d'élaboration et de simulation.

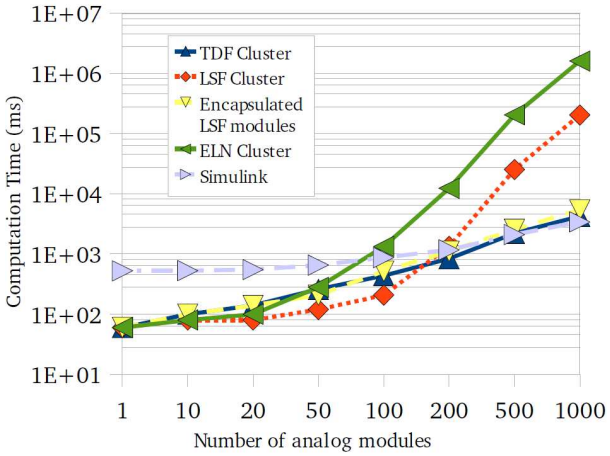


Figure 12. Durées d'élaboration et de simulation en fonction de l'ordre du filtre

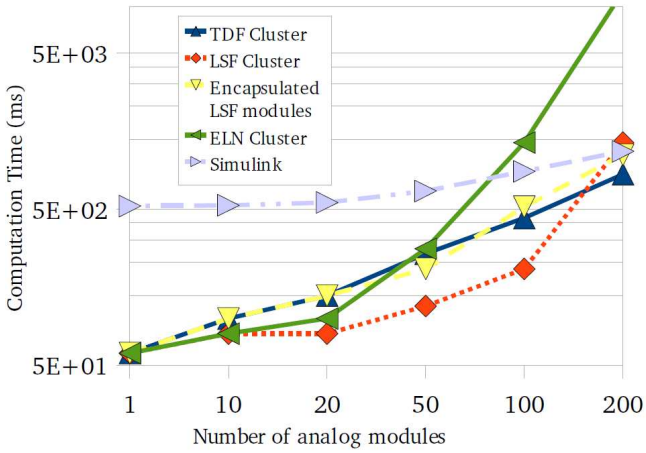


Figure 13. Durées d'élaboration et de simulation (zoom)

Franck Paugnat [Paugnat 2012] a poursuivi ces travaux sur la version 1.0 bêta 2 de SystemC AMS. Les résultats n'ont pas fait état de changement significatif. Par contre, il a analysé et séparé les temps d'élaboration et de simulation proprement dite. Les tests ont montré que les temps de simulation des modèles ELN et LSF étaient plus courts que ceux des modèles TDF et LSF encapsulé. Ceci signifie que c'est la phase d'élaboration qui est très coûteuse en temps pour les modèles ELN et LSF. L'analyse nodale modifiée est basée sur des matrices permettant la résolution du système d'équations. La taille des matrices dépend du nombre de nœuds du circuit (ELN) ou du nombre de signaux (LSF) et aussi du nombre de primitives introduisant des équations supplémentaires (`sca_eln::sca_c` ou `sca_lsf::sca_ltf_nd` en font partie). Les matrices sont souvent très creuses et des optimisations sont menées, générant de nouvelles matrices permettant d'effectuer des calculs plus rapides par la suite. Il est évident que plus la taille des matrices initiales est grande, plus l'optimisation sera coûteuse en temps. Il n'est pas possible de réutiliser le produit de l'élaboration dans la version actuelle de SystemC AMS. Ainsi, les modèles ELN et LSF comprenant un grand nombre d'instances de primitives introduisant des équations se trouvent pénalisés. Jusqu'à une amélioration de SystemC AMS permettant de réutiliser le produit de l'élaboration, nous préconisons donc, pour les modèles très lourds, d'utiliser des modèles LSF encapsulé. Ceci permet de réduire les temps de simulation à des durées proches de ceux des modèles TDF.

3.2.2 Module unique mais plus complexe

Dans cette partie, nous allons analyser le cas d'un système instancié une seule fois mais qui est plus complexe qu'une simple fonction de transfert. Le cas d'étude, présenté en Figure 14, est un four dont la température est contrôlée par un régulateur PI (proportionnel – intégral). Le système est constitué de deux parties : une partie contrôle qui permet de générer la commande en fonction de la consigne et du signal généré par le capteur et l'unité de cuisson constituée de la cavité équipée d'un corps de chauffe (résistance) et du capteur de température. La température de la cavité ne peut pas varier instantanément du fait de l'inertie

thermique, elle est donc modélisée par un comportement de type passe-bas. L'ouverture de la porte du four est modélisée par une chute de la température de la cavité.

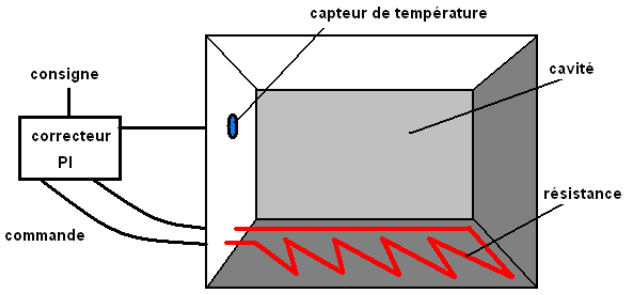


Figure 14. Modèle du Four

3.2.2.1 Modèle TDF

Le modèle TDF du four est présenté en Figure 15. Il est constitué de deux fonctions de transfert `sca_tdf::sca_ltf_nd`. La consigne est maintenue constante tout au long de la simulation. La première fonction de transfert permet de modéliser le régulateur PI. La seconde modélise la cavité. En cours de simulation, ses coefficients sont modifiés afin de modéliser l'ouverture de la porte du four puis, ils sont de nouveau modifiés afin de reprendre leurs valeurs initiales. Ce modèle introduit 2 solveurs et 3 équations.

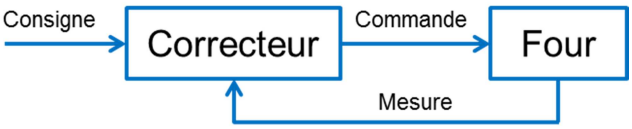


Figure 15. Modèle TDF du four

3.2.2.2 Modèle LSF

Le modèle LSF est présenté par la Figure 16. La structure interne du régulateur n'a pas été modélisée par une fonction de transfert mais nous avons opté pour un assemblage de type schéma-bloc avec un bloc de gain `sca_lsf::sca_gain` (correction proportionnelle) et un bloc intégrateur `sca_lsf::sca_integ` (correction intégrale) dont les signaux de sortie sont ensuite ajoutés *via* un additionneur `sca_lsf::sca_add`. La consigne est générée par une source LSF `sca_lsf::sca_source` et maintenue constante. Pour modéliser la cavité, nous avons dû créer notre propre primitive LSF. En effet, il n'est pas possible de modifier les coefficients de la classe `sca_lsf::sca_ltf_nd` en cours de simulation. Nous avons donc modélisé la cavité par une équation différentielle du premier ordre dont on vient modifier les coefficients au cours de la simulation afin de simuler l'ouverture de la porte. Dans ce modèle, chaque signal LSF contribue pour une équation. Les primitives n'introduisent aucune équation. Ainsi, 6 équations et un unique solveur apparaissent.

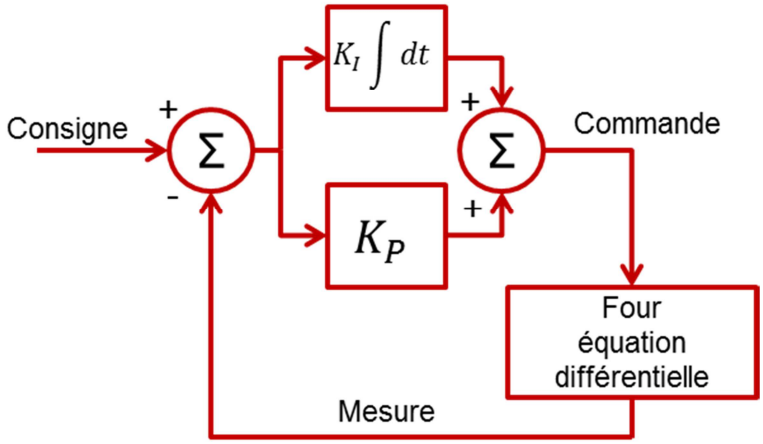


Figure 16. Modèle LSF du four

3.2.2.3 Modèle ELN

L'idée est ici de modéliser un système qui est hétérogène puisqu'il comporte une partie électrique analogique, le régulateur, et une partie thermique - le four – en utilisant le MoC ELN. Ainsi, nous introduisons dès maintenant la modélisation multiphysique sur un exemple simple. Le modèle est entièrement réalisé en ELN. La Figure 17 présente le modèle ELN du four. La consigne est constante durant la simulation et générée par une source de tension `sca_eln::sca_vsource`. Le régulateur est modélisé au moyen d'un circuit basé sur des AOP. Le schéma bloc présenté en Figure 16 est ici raffiné afin de déterminer le circuit électrique. L'architecture haut niveau du régulateur PI est donc la même en ELN qu'en LSF. La mesure est comparée à la commande au moyen d'un soustracteur. Le signal généré par le soustracteur est amplifié d'une part et intégré d'autre part. Le signal amplifié et le signal intégré sont ensuite additionnés afin de générer la commande. L'unité de cuisson est modélisée par un filtre passe bas RC. Une résistance variable sert à modéliser l'ouverture de la porte du four. Ce circuit contient 12 nœuds, une source de tension, 2 capacités et 4 nullors : 19 équations sont donc introduites. Les résistances n'introduisent aucune équation. En plus du solveur ELN, un solveur TDF est instancié puisqu'une résistance est variable (`sca_eln::sca_tdf_r`) donc pilotée par un signal TDF.

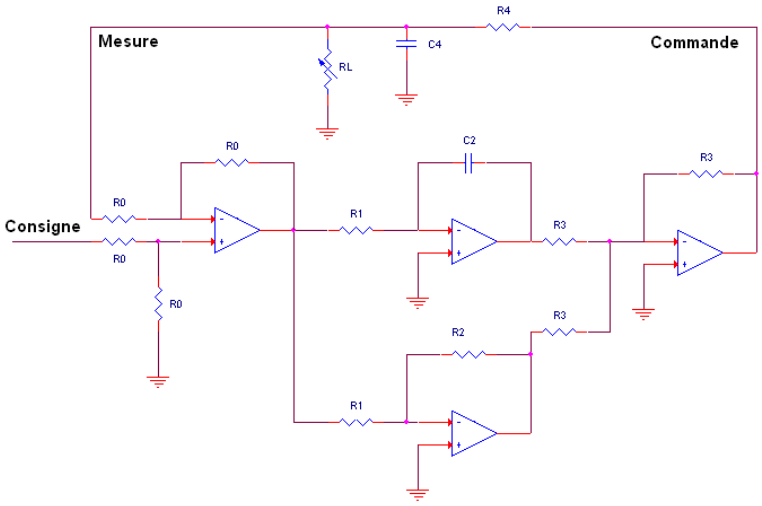


Figure 17. Modèle ELN du four

3.2.2.4 Modèle LSF encapsulé

Le modèle LSF encapsulé se présente sous la même forme que le modèle LSF. Il est bâti autour des mêmes primitives LSF. Celles-ci sont précédées et suivies de convertisseurs TDF vers LSF et LSF vers TDF, respectivement. Ceci permet de limiter le nombre d'équations par solveur. Le modèle, présenté en Figure 18 compte 5 primitives LSF, auxquelles il faut ajouter la source, 6 solveurs LSF seront donc instanciés. Les signaux LSF sont au nombre de 13 : 1 pour la source, 3 pour le soustracteur, 2 pour l'intégrateur, 2 pour l'amplificateur (gain), 3 pour l'additionneur et 2 pour le modèle de la cavité. Un septième solveur, TDF cette fois est instancié, il sert de conteneur à l'ensemble.

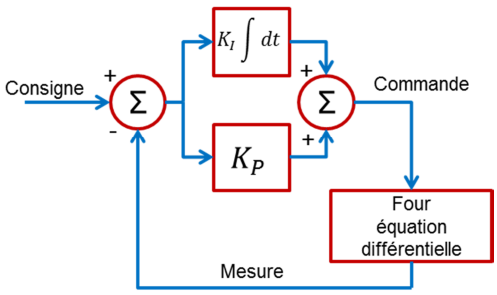


Figure 18. Modèle LSF encapsulé du four

3.2.2.5 Module unique mais plus complexe – Résultats

Le Tableau 5 présente un récapitulatif du nombre d'équations générées et du nombre de solveurs instanciés suivant le modèle. A partir de ces données, on peut prévoir que le modèle le plus rapide à simuler (simulation + élaboration) sera le modèle TDF. D'autre part, le modèle le plus lent devrait être le modèle ELN.

Tableau 5. Récapitulatif des styles de modélisation utilisés – Module unique

Style de modélisation	Nombre total d'équations générées	Nombre total de solveurs instanciés
TDF	3	2
LSF	6	1
ELN	19	2
LSF encapsulé	13	7

Les simulations ont été réalisés avec la même machine que pour le passage à l'échelle, à savoir un CPU Intel Xeon X5570 (2.93 Ghz, 8 Mo), RAM 24 Go (1066 MHz), HDD 10000 tours/min. La version 1.0 bêta 1 de SystemC AMS, pour SystemC c'était la version 2.2. Les paramètres de la simulation sont donnés dans le Tableau 6.

Tableau 6. Paramètres de la simulation – Module unique

Durée simulée	**10 s**
Pas de temps du simulateur	**100 µs**
Résolution temporelle du simulateur	**100 µs**

Les durées de simulation (élaboration + simulation) apparaissant dans la Figure 19 sont données en ms et sont des valeurs moyennes calculées à partir de 10 essais. Dans le cas du four, il est raisonnable de considérer le temps d'élaboration comme étant négligeable par rapport au temps de simulation. En effet, le modèle ELN introduit 18 équations et c'est à partir d'une vingtaine d'équations que le temps d'élaboration d'un modèle ELN commence à croitre, puis à partir d'une trentaine d'équations qu'il devient non négligeable.

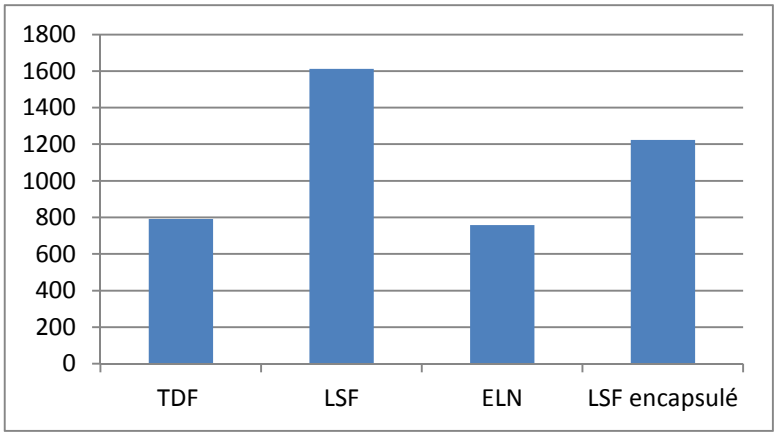


Figure 19. Durées de simulation du modèle de four suivant le style de modélisation

Nous nous apercevons que le style de modélisation basé sur ELN est légèrement plus rapide que le modèle TDF (de l'ordre de 3 %). Ainsi, résoudre un modèle ELN introduisant une vingtaine d'équations est équivalent à résoudre un modèle TDF basé sur deux fonctions de transfert d'ordre 1.

Franck Paugnat a poursuivi ces explorations en comparant ces résultats avec ceux fournis par un modèle Simulink, puis en reproduisant ces tests sur la version suivante de SystemC AMS : la version 1.0 bêta 2. Trois modèles Simulink ont été construits. Dans les trois cas, l'unité de cuisson est modélisée par une fonction de transfert. C'est dans la manière de modéliser le régulateur que les modèles diffèrent. Un premier modèle a été construit à partir du bloc régulateur PID fourni par la bibliothèque Simulink. Le second modèle utilise une fonction de transfert, ceci est équivalent au modèle TDF. Le troisième modèle utilise des blocs d'amplification et d'intégration, ce qui équivaut au modèle LSF. Tout d'abord, il apparaît clairement que les modèles Simulink du four produisent des durées de simulation 20 fois plus importantes que le modèle TDF construit à partir de SystemC AMS. La simulation du modèle Simulink équivalent au modèle LSF dure légèrement moins longtemps que les deux autres, qui sont égales. Un résultat intéressant sur la version 1.0 bêta 2 de SystemC AMS est que le modèle LSF voyait sa vitesse de simulation considérablement augmentée. En effet, dans la version bêta 1, les boucles de rétroaction ne fonctionnaient pas de façon optimale. Ce point ayant été corrigé et amélioré pour la version bêta 2 de SystemC AMS, le modèle LSF devient le plus rapide de tous, le modèle LSF encapsulé devenant le plus lent.

3.3 Conclusion du chapitre

Un choix pertinent du style de modélisation SystemC AMS repose essentiellement sur deux données : la taille du système d'équation engendré par le modèle et l'amleur de la simulation à réaliser (nombre d'échantillons à évaluer). Ainsi, le programmeur devra être capable

d'estimer le nombre de modules TDF, le nombre de signaux LSF ou le nombre de nœuds ELN. En dessous d'un certain nombre d'équations (une cinquantaine pour ELN et entre 100 et 200 pour LSF) les styles ELN et LSF sont à privilégier si la simulation est importante. Dans ce cas la durée de l'élaboration reste négligeable par rapport à la durée de la simulation. Dans l'autre cas, si le nombre d'échantillons à évaluer est faible, TDF est préférable à ELN ou LSF car les temps d'élaborations deviennent non négligeables par rapport à la durée de la simulation à proprement parler.

Si le système d'équations est important, le style TDF est le plus performant. Il sera équivalent à Simulink, mais les possibilités offertes par SystemC AMS (raffinement, interfaçage simple avec du numérique pur – SystemC, interfaçage avec des blocs plus détaillés - ELN ou LSF) en font un choix plus attrayant. Si le nombre d'échantillons à évaluer est faible, les modèles ELN et LSF sont encore plus lents.

Enfin, le style LSF encapsulé a vu son apport diminuer avec la version1.0 bêta 2 de SystemC AMS. Il reste néanmoins intéressant dans le cas de modèles très importants à simuler sur un petit nombre de points. La Figure 20 présente les styles de modélisation les plus adaptés, selon nous, aux différentes situations. Ce travail a fait l'objet d'une publication [Paugnat 11].

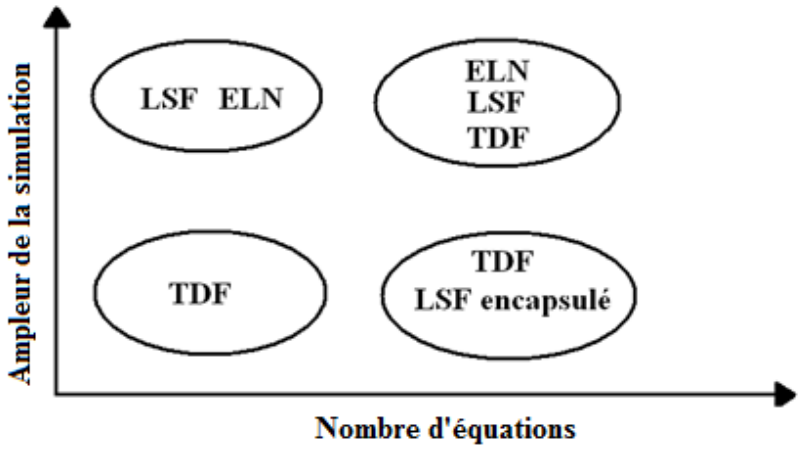


Figure 20. Performances MoCs en fonction de la taille du modèle et de l'ampleur de la simulation

4. Génération automatique de modèles de haut-niveau enrichis pour les circuits linéaires

Le travail présenté au chapitre précédent a fait ressortir que SystemC AMS permet de travailler à différents niveaux d'abstraction ainsi qu'en utilisant différents styles de modélisation. Les résultats ont montré que les simulations menées avec les styles LSF et ELN sont les plus longues, mais ceci en prenant en compte le temps d'élaboration. Dans les prochaines versions de SystemC AMS, nous pouvons espérer pouvoir réutiliser le résultat de l'élaboration, ce seront alors les styles LSF et ELN qui seront les plus rapides. D'autre part, il est apparu que les simulations de modèles LSF sont plus rapides que les simulations de modèles ELN équivalents : il y a le même nombre de solveurs mais moins d'équations, un style étant conservatif (ELN) et l'autre non conservatif (LSF). L'idée est donc ici de développer une méthode permettant de passer automatiquement d'un modèle ELN ou d'un équivalent conservatif à un modèle LSF. Ce chapitre présente une méthode automatique permettant d'obtenir une représentation de haut niveau d'un bloc analogique en ayant comme point de départ un modèle de bas niveau de type netlist, puis comment enrichir le modèle de haut niveau généré avec des informations permettant d'estimer la consommation.

Nous avons défini précédemment le flot de conception des systèmes hétérogènes. Un des avantages de la modélisation et la simulation à haut niveau d'abstraction est que la durée des simulations est considérablement réduite. D'autre part, l'outil SystemC AMS permet de modéliser toutes les composantes de ces systèmes : partie logicielle, numérique, RF et MEMS. Ainsi, le comportement global d'un système hétérogène peut être modélisé et simulé sous un environnement unique, et les blocs peuvent être raffinés un à un en remplaçant le modèle de haut niveau par un autre plus détaillé. De cette façon, le concepteur peut se livrer à une exploration architecturale méthodique. Une fois que l'on a pu déterminer pour un bloc une architecture convenable en termes de spécifications, ce bloc est de nouveau exprimé à haut niveau, afin de pouvoir maintenir des durées de simulation réduites tout au long du raffinement du système. Cependant, plutôt que d'utiliser le modèle de haut niveau de départ, nous préconisons d'utiliser les résultats de l'étape de raffinement afin de générer un modèle de haut niveau plus riche, incorporant par exemple les données nécessaires au calcul de l'énergie consommée par ce bloc. Ces informations peuvent être cruciales lorsqu'il s'agira de raffiner d'autres blocs. Cela pourra également permettre d'estimer très tôt dans le flot de conception la consommation globale du système. De plus, ces informations n'alourdissent que très peu le modèle et ne seront donc que peu coûteuses en termes de durée de simulation. Nous proposons une méthode pour les parties analogiques que nous allons présenter dans les paragraphes suivants. Cette méthode est illustrée par la Figure 21.

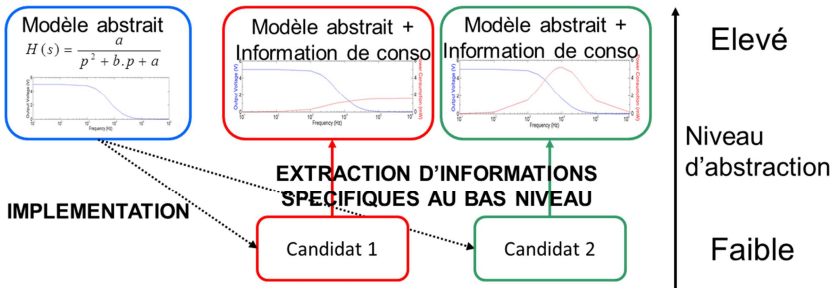


Figure 21. Méthode de raffinement et d'abstraction avec enrichissement

Le modèle abstrait de départ (en bleu) est une fonction de transfert issue des spécifications. Le modèle abstrait obtenu après application de la méthode (rouge ou vert) est une représentation d'état. L'intérêt de passer d'une fonction de transfert à une représentation d'état est que la représentation d'état est plus riche en termes d'informations contenues. En effet, une fonction de transfert H_1 permet d'exprimer un rapport, en général la sortie d'un système sur son entrée. Si l'on veut exprimer, toujours en fonction de l'entrée, un autre signal que la sortie du système, il faut une autre fonction de transfert H_2. Dans la représentation d'état, le modèle matriciel permet d'exprimer le signal à n'importe quel point du système à partir des entrées et des variables d'état.

4.1 Raffinement d'une fonction de transfert

4.1.1 Méthode de raffinement

Ce paragraphe présente la première étape : le raffinement. En partant d'une fonction de transfert, il est possible de générer automatiquement un circuit ayant un comportement correspondant. A une fonction de transfert donnée, il existe même plusieurs architectures correspondantes. Ainsi, la fonction de transfert n'est pas suffisante pour débuter le processus de raffinement, il faut aussi spécifier un type d'architecture. Dans le cas de filtres analogiques, on spécifiera le type de cellule de base utilisée pour le raffinement : cela pourra être la cellule de base d'un filtre actif (comportant un AOP) comme la cellule de Rauch ou celle de Sallen-Key. On pourra aussi utiliser des circuits passifs, dans ce cas il faudra spécifier si les composants sont associés en T ou en π par exemple. Il est à noter que des méthodes automatiques bien plus performantes que la nôtre peuvent être trouvées, certains programmes en ligne proposent de spécifier le gabarit et la structure de base et renvoient les valeurs des composants, par exemple sur [CalcEdge]. Toutefois, il nous est apparu utile de disposer d'un outil, aussi simple soit-il permettant de générer automatiquement des modèles électriques de filtres (ELN) sous SystemC AMS.

Le point de départ de la méthode est la spécification du filtre : le gabarit. Deux points (au moins - dans le cas d'un filtre passe-bande ou d'un filtre coupe-bande, il faudra définir quatre points du gabarit) du diagramme de gain (Bode) sont précisés. Le premier correspond à l'atténuation maximale en bande passante (A_{MAX}) et à la fréquence correspondante (F_1). Le

second correspond quant à lui à l'atténuation minimale en bande coupée (A_{MIN}) et à la fréquence correspondante (F_2). Matlab est capable de fournir à partir de ces deux points l'ordre du filtre ainsi que sa fréquence de coupure en utilisant les fonctions `buttord` et `butter`, pour la méthode de Butterworth. Voici un exemple d'utilisation de cette fonction :

```
[n,Wn] = buttord (314e3,628e3,6,40,'s')
```

`n` et `Wn` sont respectivement l'ordre du filtre et la pulsation de coupure. Les deux premiers paramètres de la fonction `buttord` correspondent aux fréquences F_1 et F_2. Les deux paramètres suivants sont les atténuations A_{MAX} et A_{MIN}. Enfin, `s` signifie que c'est un filtre analogique.

La seconde fonction, `butter`, permet de générer la fonction de transfert, autrement dit le numérateur et le dénominateur. Voici comment cette fonction et utilisée:

```
[num,den] = butter (n,Wn,'low','s')
```

`num` et `den` sont des vecteurs, ils correspondent au numérateur et au dénominateur de la fonction de transfert. Les paramètres `n` et `Wn` de la fonction `butter` sont les résultats de l'exécution de la fonction `buttord`, soit l'ordre du filtre et sa fréquence de coupure. Enfin `low` et `s` précisent que c'est un filtre passe-bas analogique.

Il est possible d'exprimer la fonction de transfert en représentation zéros-pôles. Les zéros et les pôles sont écrits dans un fichier texte qui sera un paramètre pour la classe C++ que nous allons présenter par la suite et qui permet de générer automatiquement le circuit correspondant à la fonction de transfert. C'était la dernière étape de la procédure implémentée sous Matlab.

Nous proposons ici de synthétiser les filtres au moyen de cellules de Sallen-Key. Ce sont des cellules d'ordre 2. Ainsi, sous SystemC AMS, la première étape consiste à multiplier les pôles complexes conjugués deux à deux et à développer ce produit. Nous nous retrouvons ainsi avec un produit de polynômes du second ordre au dénominateur de la fonction de transfert. Chacune de ces fonctions de transfert d'ordre 2 permettra d'obtenir une cellule de Sallen-Key. Si la fonction de transfert de départ est impaire, le pôle unique résultant est stocké à part et sera synthétisé par un premier ordre de type RC. La Figure 22 montre une cellule active de Sallen Key, cette cellule réalise un filtre passe bas du second ordre.

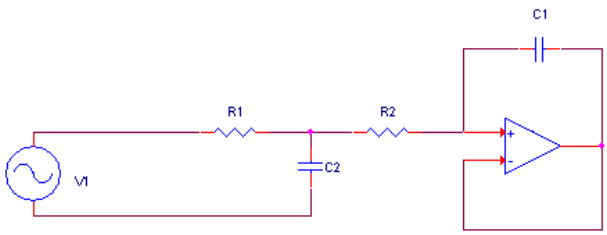


Figure 22. Schéma d'une cellule de Sallen Key du second ordre

La fonction de transfert de cette cellule est donnée par l'équation (1). Dans ce cas, le numérateur est égal au terme constant du dénominateur. Pour une cellule passe haut, on retrouvera au numérateur le terme en p^2 et pour une cellule passe bande le terme en p. A partir de la fonction de transfert on peut procéder par identification et déterminer les valeurs des composants passifs. Pour cela, il suffit de fixer en dur les valeurs des résistances à une certaine valeur (1 KΩ, nous ne nous préoccupons pas des problèmes de courant sur les entrées de l'AOP) et un calcul simple nous donnera les valeurs des capacités.

$$\frac{V_{OUT}}{V_{IN}} = \frac{\frac{1}{R_1 R_2 C_1 C_2}}{p^2 + p(R_1 C_2 + R_2 C_2) + \frac{1}{R_1 R_2 C_1 C_2}} \quad (1)$$

Nous allons prendre un exemple, soit le gabarit suivant :

- Maximum 6 dB d'atténuation en dessous de 50 KHz.
- Minimum 40 dB d'atténuation au-dessus de 100 KHz.

Soit le dénominateur d'une fonction de transfert passe bas défini dans le Tableau 7. Le numérateur de cette fonction de transfert est égal au terme constant du dénominateur puisque c'est une fonction de transfert passe bas.

Tableau 7. Fonction de transfert passe bas d'ordre 6

Terme	Valeur du coefficient
constant	$613{,}4506.10^{30}$
p	$8{,}1312.10^{27}$
$\mathbf{p^2}$	$53{,}8886.10^{21}$
$\mathbf{p^3}$	$226{,}4191.10^{15}$
$\mathbf{p^4}$	$634{,}2163.10^{9}$
$\mathbf{p^5}$	$1{,}1262.10^{6}$
$\mathbf{p^6}$	1

La fonction de transfert d'ordre 6 du Tableau 7 peut être décomposée en un produit de trois fonctions de transfert d'ordre 2. Les trois dénominateurs associés sont détaillés dans le Tableau 8.

Tableau 8. Décomposition du dénominateur

Terme	Dénominateur A	Dénominateur B	Dénominateur C
constant	$8{,}5.10^{10}$	$8{,}5.10^{10}$	$8{,}5.10^{10}$
p	$1{,}51.10^{5}$	$5{,}63.10^{5}$	$4{,}12.10^{5}$
p2	1	1	1

Après avoir appliqué la méthode d'identification et fixé les valeurs des résistances à 1 KΩ, nous obtenons les trois cellules de base présentées dans le Tableau 9.

Tableau 9. Valeur des composants passifs du filtre

Cellule	Composant	Valeur
A	R11, R12	1 KΩ
	C11	13,3 nF
	C12	0,888 nF
B	R21, R22	1 KΩ
	C21	3,55 nF
	C22	3,31 nF
C	R31, R32	1 KΩ
	C31	4,85 nF
	C32	2,43 nF

La dernière étape de la procédure consiste à générer automatiquement le circuit. Le nombre de cellules nécessaires est défini par l'ordre de la fonction de transfert. Le type des cellules (passe-bas, passe-haut...) est quant à lui défini par le numérateur. Ainsi, le circuit peut être instancié très rapidement, les valeurs des composants passifs étant ensuite calculées au cours de la phase d'identification. La Figure 23 présente le filtre global.

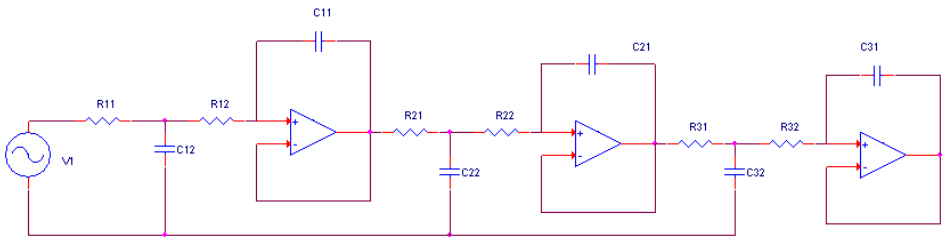


Figure 23. Filtre réalisant la fonction de transfert définie par le Tableau 7

La Figure 24 présente un résultat de simulation. Trois cellules de Sallen-Key passe bas ont été instanciées en série avec pour valeurs des composants passifs celles présentées dans le Tableau 9. Nous avons effectué une comparaison entre le modèle électrique (ELN) utilisant le nullor en tant que modèle d'amplificateur opérationnel et la fonction de transfert générée par Matlab et implémentée sous SystemC AMS au moyen du MoC LSF. Nous présentons ici le diagramme de gain des deux modèles. Les deux courbes sont superposées, ceci montre que le modèle de bas niveau correspond bien à la fonction de transfert de départ. D'autre part, les deux points spécifiés par le gabarit sont respectés, à savoir : au maximum 6 dB d'atténuation pour les fréquences inférieures à 50 KHz et au minimum 40 dB d'atténuation pour les fréquences supérieures à 100 KHz.

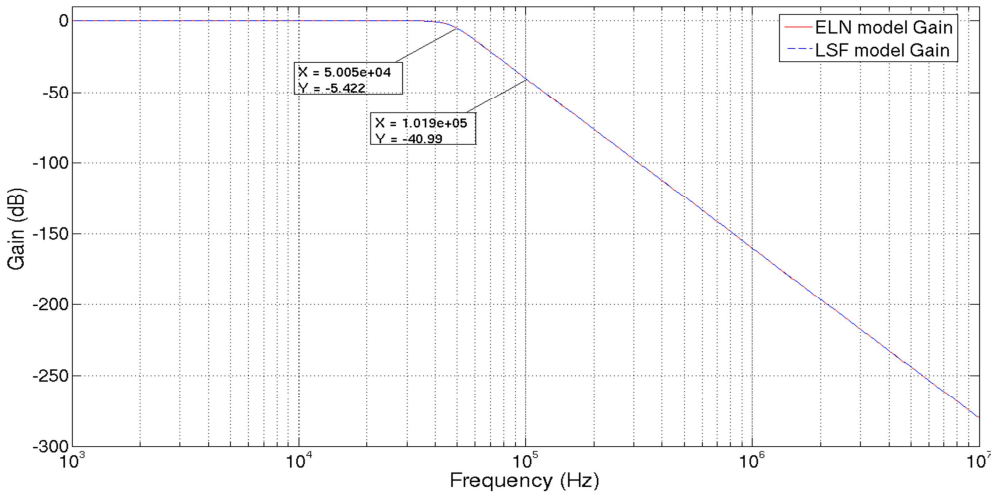


Figure 24. Digramme de gain du circuit comparé à la fonction de transfert de départ.

Il est à noter que la durée de la simulation est plus importante de l'ordre de 10 % pour le circuit ELN que pour la fonction de transfert LSF, ceci en accord avec [Paugnat 11].

Nous avons présenté ici le cas de la synthèse des filtres au moyen de cellules de Sallen-Key. Cependant, il est possible de réaliser la même opération en utilisant d'autres types de cellules de base, par exemple les cellules de Rauch. La différence réside au niveau de la phase d'identification : au lieu d'utiliser la fonction de transfert normalisée de la cellule de Sallen-

Key, il faut utiliser celle de Rauch. Notre programme offre cette possibilité, l'utilisateur devant spécifier en paramètre de la classe C++ réalisant la synthèse quel type de cellule doit être utilisé en vue de la synthèse. Par contre, en ce qui concerne la synthèse au moyen de filtres passifs, un problème d'adaptation d'impédance va se poser. Ce problème n'apparaît pas avec les filtres actifs qui sont construits autour d'un AOP (qui dispose idéalement d'une impédance d'entrée infinie et d'une impédance de sortie nulle). Ainsi, si l'ordre du filtre est élevé et que l'on souhaite le synthétiser par une association en cascade de cellules en π par exemple, il faudrait disposer un montage suiveur entre chaque étage afin de pouvoir considérer chaque filtre comme étant non chargé. Ceci permet de simplifier considérablement les calculs. Nous n'avons pas développé de méthode spécifique permettant la synthèse de filtres passifs, celles-ci existant déjà. Nous disposions d'une méthode permettant de générer des filtres actifs, ce qui était suffisant pour nous permettre d'aller suffisamment loin dans la méthode d'abstraction que nous allons présenter par la suite.

4.1.2 Implémentation sous SystemC AMS de la méthode de raffinement

La méthode présentée précédemment a été implémentée sous SystemC AMS. La classe C++ développée permet de générer automatiquement un circuit électrique ELN en lieu et place d'un bloc fonction de transfert TDF ou LSF. Nous ne détaillerons pas toutes les fonctions mais seulement celles qui sont les plus pertinentes ou importantes.

Commençons par la brique de base : la cellule de Sallen-Key. Le code présenté en Annexe 1.1 montre une cellule de Sallen-Key passe bas implémentée en SystemC AMS. Elle dispose d'un port ELN en entrée et d'un port ELN en sortie. Pour les composants passifs, nous avons utilisé les types dont la valeur est fixée par un signal de contrôle, ainsi la cellule de base dispose de 4 signaux numériques d'entrée permettant de fixer les valeurs des 4 composants passifs. Il faut aussi définir un nœud de référence et deux nœuds intermédiaires. C'est à l'intérieur du constructeur de classe que se font les instanciations ainsi que le « câblage » des composants ayant été préalablement déclarés.

Voyons maintenant la fonction permettant de calculer les valeurs des composants passifs. Le code de l'annexe 1.2 présente ce calcul pour un filtre passe bas. Le nombre de cellules d'ordre 2 (`ncells`) a été déterminé au moyen d'une autre fonction. `qp(i)` et `wp(i)`, calculés par une autre fonction afin de mettre sous forme canonique les fonctions de transfert correspondent respectivement au facteur de qualité et à la pulsation de coupure de la fonction de transfert d'indice *i*. Les vecteurs `r1`, `r2`, `c1` et `c2` sont utilisés pour stocker les résultats. Une boucle permet de balayer toutes les fonctions de transfert afin de calculer les valeurs des composants passifs de chaque cellule.

Enfin, le code présenté dans l'annexe 1.3 montre la génération d'un filtre passe bas d'ordre impair. Tout d'abord en bleu clair apparaît l'instanciation du filtre du premier ordre, il est placé en amont des cellules du second ordre. En rouge, nous retrouvons l'instanciation des cellules du second ordre avec le nom de chaque objet ainsi que les valeurs des composants

passifs. En bleu foncé apparaît la connexion de l'entrée de la première cellule au port d'entrée du circuit global et la connexion de la sortie de la dernière cellule à l'entrée du filtre d'ordre 1. Enfin, nous voyons en vert comment sont connectés les cellules entres elles.

4.2 Abstraction d'un circuit : génération automatique de la représentation d'état

La partie précédente présentait une méthode de raffinement. Elle a permis de générer un circuit réalisant les spécifications (gabarit) en passant par la fonction de transfert. Nous allons maintenant présenter une méthode permettant, à partir d'un circuit, sa netlist ou son modèle ELN, de générer une représentation d'état.

Un circuit électrique peut être décrit par un système d'équations. Chaque composant introduit au moins une équation. Les composants non linéaires peuvent être modélisés par plusieurs éléments simples. Une diode peut par exemple être modélisée par l'association série d'un interrupteur, d'une résistance et d'un générateur de tension. Une autre possibilité est de la modéliser à partir d'une résistance variable et d'un générateur de tension : si la tension à ses bornes est supérieur à un certain seuil (0,7 V dans la plupart des cas) alors la valeur de la résistance est très faible, sinon elle est très grande. Une approche classique pour la description d'un circuit sous forme matricielle consiste à utiliser l'Analyse Nodale (Nodal Analysis, NA). Le simulateur Spice utilise lui une adaptation appelée Analyse Nodale Modifiée (MNA) [Litovski 97].

4.2.1 Analyse Nodale

La topologie d'un circuit électrique se définit par une succession de nœuds et de branches. Les nœuds réalisent les connexions entre les branches alors que les branches « portent » les composants et sont connectées à un nœud à chacune de leurs deux extrémités. A chaque nœud correspond un potentiel électrique et chaque branche est traversée par un courant. L'analyse nodale consiste à exprimer les relations courant-tension de chaque branche au moyen des lois de Kirchoff et de la loi d'Ohm. La Figure 25 montre une branche résistive (à gauche) et une source de courant (à droite).

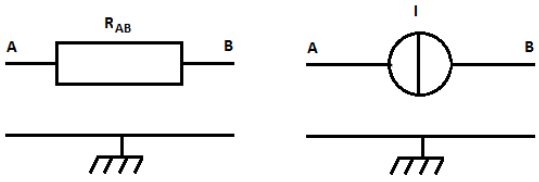


Figure 25. Une branche résistive et une source de courant

L'équation (2) régit la branche résistive, avec $G_{AB}=1/R_{AB}$. G_{AB} est l'admittance exprimée en Siemens.

$$G_{AB}.(V_A - V_B) = I_{AB} \qquad (2)$$

L'équation (3) régit la source de courant.

$$I = I_{AB} \tag{3}$$

Pour appliquer l'Analyse Nodale à un circuit complet, il faut établir la loi de fonctionnement de chacun des composants. Ainsi, un système d'équations est construit, que l'on peut exprimer sous forme matricielle, comme présenté par l'équation (4).

$$\begin{pmatrix} G_{AB} & -G_{AB} \\ -G_{AB} & G_{AB} \end{pmatrix} \cdot \begin{pmatrix} V_A \\ V_B \end{pmatrix} = \begin{pmatrix} I_{AB} \\ I_{AB} \end{pmatrix} \tag{4}$$

Chaque ligne du système matriciel correspond à un nœud, plus exactement au potentiel d'un nœud (V_N). Le nœud de référence du circuit n'étant pas pris en compte, la dimension des matrices est égale au nombre de nœuds du circuit moins un. L'inconvénient de cette méthode est que le fonctionnement d'un composant doit être décrit à partir de son admittance et du courant qui le traverse. Ainsi, une source de tension réelle doit être transformée en son équivalent source de courant alors qu'il est impossible de prendre en compte une source de tension parfaite [Legrand 04]. Pour remédier à ce problème limitant le nombre de circuits que cette méthode permet de traiter (tous ceux comportant une source de tension parfaite), il a fallu introduire l'Analyse Nodale Modifiée [Chung 75].

4.2.2 Analyse Nodale Modifiée

Comme nous l'avons vu, l'Analyse Nodale ne permet pas de décrire efficacement des sources de tension. Une source de tension parfaite est décrite par l'équation (5).

$$U_{AB} = V_A - V_B \tag{5}$$

Aucune variable de courant n'apparaît dans cette équation, on ne pourra donc pas poser la loi des nœuds et ainsi établir le système d'équations de façon correcte. Cependant, il est possible de modéliser une source de tension réelle en la transformant en source de courant réelle. Ainsi, en introduisant un élément appelé gyrateur idéal, nous allons pouvoir traiter par le biais de l'Analyse Nodale Modifiée un plus grand nombre de circuits. La Figure 26 présente le schéma du gyrateur idéal.

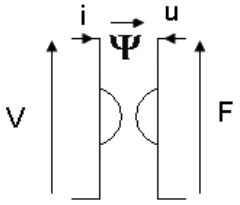


Figure 26. Schéma du gyrateur idéal

Les équations (6) et (7) permettent de décrire le fonctionnement du gyrateur idéal.

$$-V - \Psi . u = 0 \tag{6}$$

$$F - \Psi . i = 0 \tag{7}$$

Cet élément permet - s'il est couplé à une source de tension parfaite et si le facteur ψ est égal à 1 – de pouvoir traiter les circuits incluant des sources de tension parfaites. Le système matriciel devient alors tel que l'équation (8) le fait apparaître.

$$A . x = z \tag{8}$$

Avec : $A = \begin{pmatrix} G & B \\ C & D \end{pmatrix}$ $x = \begin{pmatrix} v \\ j \end{pmatrix}$ $z = \begin{pmatrix} i \\ e \end{pmatrix}$

Dans cette équation, *G*, *v* et *i* sont les éléments donnés par l'Analyse Nodale simple, à savoir la matrice admittance (*G*), le vecteur des potentiels (*v*) et le vecteur des courants (*i*). *j* est le vecteur des courants des branches ajoutées lors de l'intégration du ou des gyrateurs. *e* est le vecteur des excitations dues aux sources de tension. Enfin, les matrices *B*, *C* et *D* sont les matrices permettant de lier les anciennes variables de courant et de potentiel aux nouvelles variables de courant et de potentiel.

L'inconvénient de telles méthodes est que la plupart du temps en électronique, on souhaite plutôt obtenir une tension en résultat. Ici, le résultat de la méthode est un courant. Bien sûr, après quelques calculs matriciels, il serait possible d'extraire les différents potentiels du circuit. Un autre inconvénient est que l'on apprécierait d'avoir un rapport entre la sortie et l'entrée. Là aussi, l'analyse nodale et l'analyse nodale modifiée ne permettent pas cela. Enfin, ces modèles sont très détaillés, toutes les informations contenues dans le modèle de bas niveau sont contenues dans ces modèles. L'idée serait plutôt d'obtenir une représentation équivalente à la fonction de transfert, qui permette donc d'avoir une relation entre la sortie et l'entrée du système et qui soit aussi plus légère en termes d'informations afin de raccourcir les durées de simulations. L'inconvénient de la fonction de transfert est qu'elle n'exprime qu'un seul rapport en deux signaux du circuit. On peut par exemple exprimer la tension de sortie par rapport à la tension d'entrée mais si l'on souhaite exprimer le potentiel à un nœud intermédiaire en fonction de l'entrée, il faut une seconde équation.

La représentation d'état va nous permettre de palier ces différents inconvénients. D'une part, elle est potentiellement plus riche en informations que la fonction de transfert puisque chaque signal du circuit peut être inclus dans le vecteur de sortie. D'autre part, on peut très bien se limiter à un seul signal dans ce même vecteur de sortie, ce qui permet d'avoir une représentation aussi abstraite que la fonction de transfert.

4.2.3 La représentation d'état

Nous allons nous focaliser sur les systèmes linéaires dynamiques. La Figure 27 présente le diagramme d'un tel système.

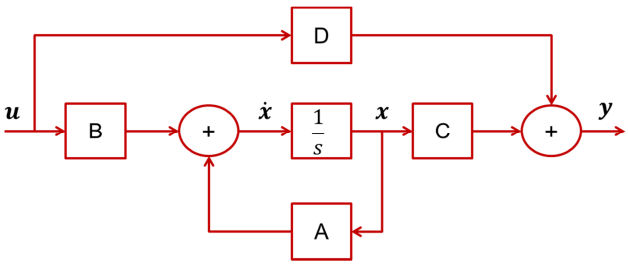


Figure 27. Diagramme schéma-bloc d'un système linéaire dynamique

Une représentation d'état d'un système linéaire dynamique est définie par deux équations : l'équation d'état ou équation dynamique (9) et l'équation de sortie (10).

$$\dot{x} = A.x + B.u \tag{9}$$

$$y = C.x + D.u \tag{10}$$

Dans les deux équations précédentes, *A*, *B*, *C* et *D* sont les matrices du système, elles sont constantes dans le temps, *u* est le vecteur d'entrée du système, *y* est le vecteur de sortie, *x* est le vecteur des variables d'état et $\dot{x}$ est le vecteur des dérivées des variables d'état. La matrice *A* est appelée matrice d'état, la matrice *B* est la matrice de commande, la matrice *C* est la matrice d'observation et la matrice *D* est la matrice d'action directe. La représentation d'état est un outil très largement utilisé par les automaticiens. En effet, à partir des quatre matrices du système on peut déterminer les propriétés de stabilité, commandabilité et observabilité du système. Cependant, nous allons utiliser cet outil uniquement dans un but de modélisation des systèmes linéaires.

Dans la suite, l'objectif est de déterminer, pour un circuit donné, l'ensemble des composantes des deux équations (9) et (10), c'est-à-dire qu'il faudra déterminer les variables d'état ainsi que les quatre matrices de la représentation d'état. Les variables d'état d'un système, $x_1(t)$, $x_2(t)$, $x_3(t)$…sont les variables dont la connaissance de la valeur à un instant t_i permettent de déterminer l'état du système à n'importe quel instant futur, si l'on connait les entrées $u_1(t)$, $u_2(t)$, $u_3(t)$… C'est un ensemble minimum de variables. Leur association constitue le vecteur d'état. En électricité, les variables d'état sont la tension aux bornes d'une capacité et le courant traversant une inductance. A partir de la représentation d'état, il est possible de trouver les fonctions de transfert associées à chacun des éléments du vecteur de sortie.

La méthode présentée dans les pages suivantes est assez proche de l'Analyse Nodale et de l'Analyse Nodale Modifiée, elle va consister à construire un système matriciel à partir des

équations décrivant le comportement de chacun des composants du circuit. La différence réside dans le fait que l'on cherche à alléger le plus possible la simulation, ainsi, le modèle final ne comportera que les informations nécessaires à la simulation d'un ou plusieurs signaux du circuit, mais pas de tous, [Simeu 99].

4.2.4 Présentation de la méthode d'abstraction des circuits analogiques linéaires

Le point de départ de notre méthode est une netlist. Une netlist est un modèle de bas niveau d'abstraction qui inclue des informations sur les composants du circuit comme leur type et leur valeur mais aussi sur l'architecture puisqu'il est décrit comment sont connectés les composants entre eux.

La première étape consiste à analyser cette netlist. En effet, chaque composant introduit une équation définissant son comportement. Par exemple, l'équation correspondant à une résistance est la loi d'Ohm. Le Tableau 10 présente les équations qui correspondent aux différents composants. On considère le second membre de chaque équation comme étant nul. Ainsi, l'amplificateur opérationnel (AOP) est considéré comme fonctionnant en régime linéaire (rétroaction négative) donc la tension différentielle d'entrée $\varepsilon = V^+ - V^-$ est considérée comme étant nulle. De plus, l'AOP est supposé idéal, donc les courants d'entrée I^+ et I^- sont considérés comme étant nuls. Ces courants ne sont donc pas ajoutés à la liste des variables du système : la branche reliant l'entrée d'un AOP à un nœud n'introduit pas d'équation.

Les variables du système sont le courant dans chaque branche ainsi que chaque potentiel électrique. L'écriture des équations se fait de la façon suivante : une matrice est créée, chaque ligne correspond à une équation et chaque colonne correspond à une variable. Ainsi, pour écrire une équation définissant le comportement d'une résistance $u_P - u_N - R.i_R = 0$, les colonnes correspondant aux variables u_p, u_n et i_r seront « remplies » avec, respectivement, les valeurs *1*, *-1* et *–R*. La même opération est reproduite pour chaque composant, l'opération introduisant une nouvelle ligne dans la matrice à chaque fois. Après avoir traité tous les composants de la netlist, il faudrait créer un vecteur correspondant au second membre et ne contenant que des zéros. Cette étape n'est pas absolument nécessaire et peut être coûteuse en ressource dans le cas de circuits comportant un très grand nombre de nœuds, elle n'est donc pas réalisée.

La seconde étape consiste à réorganiser la matrice d'une façon judicieuse. Les équations d'état (faisant intervenir une dérivée de variable d'état) doivent être placées dans les lignes supérieures de la matrice. Parallèlement, les variables d'état, leurs dérivées ainsi que les entrées doivent être placées dans les colonnes de gauche de la matrice. Ceci conduit immédiatement à identifier quatre sous matrices qui sont présentées en Figure 28.

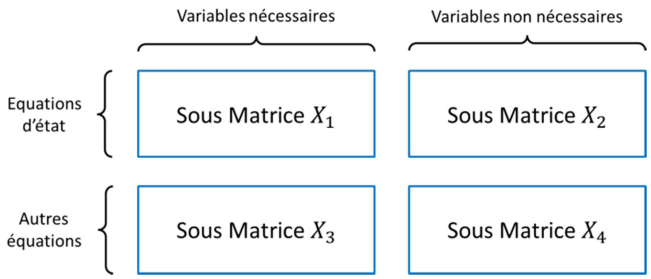


Figure 28. Les différentes sous matrices de la matrice du circuit

Tableau 10. Equations définissant le comportement des composants électriques linéaires

Composant	Equation	KCL
Résistance	$-i_R.R + (u_{N1} - u_{N2}) = 0$	$n_1 : +i_R$ $n_2 : -i_R$
Capacité	$-C.\dot{u_{N1N2}} + i_C = 0$	$n_1 : +i_C$ $n_2 : -i_C$
Inductance	$-L.\dot{i_L} + (u_{N1} - u_{N2}) = 0$	$n_1 : +i_L$ $n_2 : -i_L$
Source de tension	$-V + (u_{N1} - u_{N2}) = 0$	
Source de courant		$n_1 : +I$ $n_2 : -I$
Source de tension commandée en tension (VCVS)	$-E.(u_{N1} - u_{N2}) + (u_{N3} - u_{N4}) = 0$	
Source de tension commandée en courant (CCVS)	$-H.i_{N1N2} + (u_{N3} - u_{N4}) = 0$	
Source de courant commandée en courant (CCCS)		$n_3 : +F.i_{N1N2}$ $n_4 : -F.i_{N1N2}$
Source de courant commandée en tension (VCCS)		$n_3 : +G.(u_{N1} - u_{N2})$ $n_4 : -G.(u_{N1} - u_{N2})$
Nœud de référence	$u_0 = 0$	
Transformateur	$-V_2.X - V_1.Y = 0$ $-I_1.X - I_2.Y = 0$	
Gyrateur	$F - \Psi.i = 0$ $-V - \Psi.u = 0$	
Amplificateur opérationnel	$-V^- + V^+ = 0$	

Les lignes de la sous matrice X_1 correspondent aux équations d'état. Ses colonnes correspondent aux variables d'état, à leurs dérivées et aux entrées. C'est cette matrice qui nous permettra de générer l'équation (9) à la condition que tous les éléments de la sous matrice X_2 soient nuls. Nous appelons l'ensemble des variables de la matrice X_1 les « variables nécessaires », dans le sens où elles apparaissent soit dans le vecteur des entrées, soit dans le vecteur des variables d'état, soit dans le vecteur des dérivées des variables d'état. Au cours de la résolution partielle du système, elles doivent être gardées telles quelles, elles ne doivent pas être exprimées en fonction d'autres variables. Toutes les autres variables, appelées « variables non nécessaires » devront être exprimée en fonction des variables nécessaires au cours de la résolution partielle. La sous matrice X_2 fait aussi intervenir les équations d'état mais les variables qui lui sont associées sont les variables non nécessaires. La sous matrice X_4 doit être rendue triangulaire supérieure afin de faciliter les étapes suivantes. Cela permettra de résoudre une à une les lignes de la sous matrice X_4, en partant de la dernière ligne. Cette dernière opération est réalisée par des permutations de lignes et de colonnes sur les sous matrices X_3 et X_4. L'algorigramme correspondant aux permutations de lignes est donné en Figure 29. Le nombre de termes non nuls est uniquement calculé sur les colonnes de la sous matrice X_4. Par contre, lorsqu'une permutation est effectuée, elle s'opère sur une ligne de la matrice globale : la sous matrice X_3 voit aussi ses lignes permutées. La Figure 30 montre l'algorigramme de permutation des colonnes. Cette opération s'effectue à partir de la sous matrice X_4 sur les sous matrices X_2 et X_4. A la suite de ces deux opérations de permutation, la sous matrice X_4 est une matrice triangulaire supérieure et l'étape de réorganisation de la matrice est achevée.

L'étape suivante permet de générer une nouvelle matrice, dite matrice de relation, où les colonnes sont les variables non nécessaires et les lignes les variables nécessaires. Ainsi, au moyen de cette matrice, nous pourrons déterminer l'expression de chaque variable non nécessaire en fonction des variables nécessaires. Les termes non-nuls de la sous matrice X_2 peuvent désormais être remplacés par leur expression en fonction des variables nécessaires.

Une fois que tous les éléments de la sous matrice X_2 sont nuls, les lignes de la sous matrice X_1 correspondent à l'équation (9), l'équation d'état, ainsi les matrices *A* et *B* sont extraites directement.

Jusqu'ici, nous ne nous sommes pas préoccupés de ce que pouvaient être les composantes du vecteur de sortie de l'équation (10). Les étapes réalisées jusqu'à présent sont indépendantes des signaux que l'on veut incorporer en tant que composantes du vecteur de sortie. Le vecteur de sortie est généré suivant la convenance de l'utilisateur. Au moment de l'instanciation de la classe SystemC AMS, l'utilisateur spécifie librement les signaux qu'il veut voir intégrés au vecteur de sortie. Les signaux de sortie seront exprimés à partir de la matrice de relation et en fonction des variables nécessaires. Si l'on veut intégrer au vecteur de sortie une variable d'état ou une entrée, soit une variable nécessaire alors le résultat est évident puisque cette variable est contenue soit dans le vecteur de variables d'état ou dans le vecteur des entrées : il suffira

d'ajouter une ligne dans la matrice *C* qui contient un '1' sur la colonne correspondante, les autres composantes de la ligne étant nulle. On ajoutera dans la matrice *D*, une ligne de composantes toutes nulles au même niveau. Pour tout autre signal du circuit, notamment une variable non nécessaire, il faudra se référer à la matrice de relations et extraire la ligne correspondant à la variable non nécessaire souhaitée.

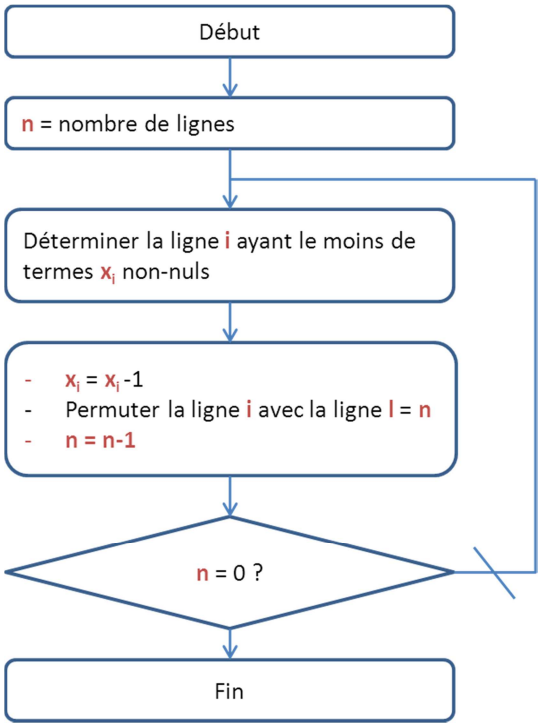


Figure 29. Algorigramme des permutations de lignes

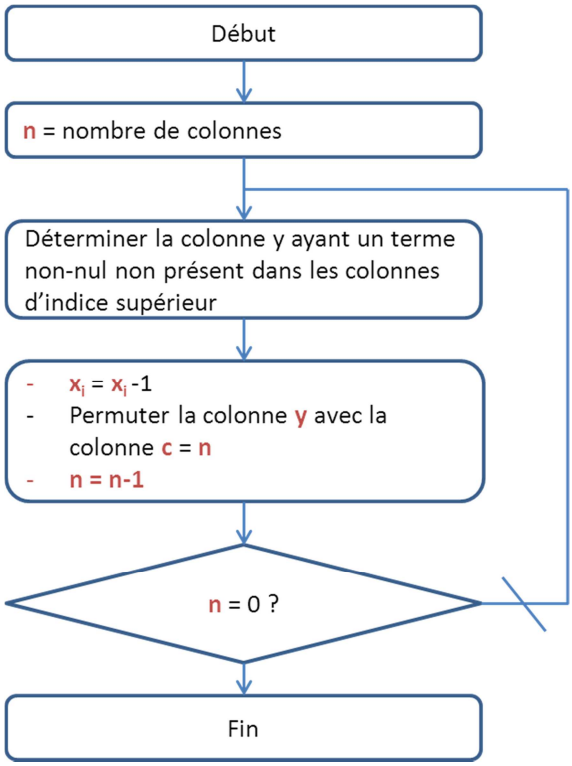


Figure 30. Algorigramme des permutations de colonnes

4.3 Cas particuliers / Améliorations

Le filtre présenté en Figure 23 fait intervenir des capacités. Celles-ci peuvent être connectées entre un nœud et le nœud de référence, la masse. Si une capacité n'est pas connectée au nœud de référence, ce qui arrive si l'on interverti R_{11} et C_{12}, il faut introduire une nouvelle équation. En effet, dans cette configuration, l'équation d'état de C_{12} est donnée par l'équation (11).

$$-C_{12}.\dot{u_{12}} + i_{C12} = 0 \qquad (11)$$

Par défaut, notre programme considère que la différence de potentiel entre un nœud N et le nœud de référence est égale au potentiel du nœud N. Ainsi, par exemple, $u_{20} = u_2$. Une équation doit être ajoutée à la matrice afin de définir $u_{12} = u_1 - u_2$. La structure du circuit peut aussi être trompeuse. Traditionnellement, le nombre d'inductances ajouté au nombre de capacités donne l'ordre du filtre ainsi que le nombre de variables d'état. Cependant, le cas présenté par la Figure 31 est problématique. En effet, hormis les trois variables d'état introduites par les trois capacités : u_{C1}, u_{C2} *et* u_{C3}, il faudrait en ajouter deux supplémentaires : les courants traversant les deux inductances i_{L1} et i_{L2}. Or, les variables d'état d'un système sont indépendantes. Ici, ce n'est pas le cas ; le transformateur permet un isolement galvanique mais les deux courants i_{L1} et i_{L2} sont liés par la relation suivante : $i_{L2} = A_1.i_{L1}$. Dans ce cas, le mieux à faire est de ramener l'inductance L_2 au primaire du transformateur et de ne considérer plus qu'une seule inductance égale à la somme de L_1 et L_2.

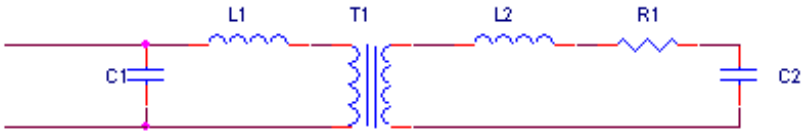


Figure 31. Exemple problématique

4.4 La classe `Netlist2ss`

La méthode présentée précédemment a été implémentée dans l'environnement SystemC AMS. Nous ne détaillerons pas ici toutes les fonctions mais seulement celles qui nous semblent les plus pertinentes ou plus importantes.

Le fichier texte contenant la netlist est passé en paramètre lors de l'instanciation de la classe. Une première fonction permet donc de lire ce fichier et d'en identifier toutes les parties. Une netlist est un fichier contenant la description d'un circuit électrique sous forme de texte. Chaque ligne correspond à un composant. Une ligne est constituée de « mots » qui sont séparés par un espace. Nous avons donc créé une structure `Component` contenant les champs suivants : `Name`, `Type`, `Node_p`, `Node_n`, `Value` et `eq_comp`. Le champ `Name` correspond au nom du composant, sa valeur est donc égale au premier mot de la ligne, c'est une chaine de caractères. Le champ `Type` correspond à la première lettre du premier mot de la ligne, c'est un caractère. `Node_p` et `Node_n` sont les numéros des nœuds auxquels sont connectés respectivement l'interface positive et l'interface négative d'un dipôle (nous reviendrons plus tard sur les composants qui ne sont pas des dipôles). Ces champs sont de type nombre entier et sont identifiés par le deuxième et le troisième mot de la ligne. Le champ `Value`, de type réel double, contient la valeur du composant. Il est identifié par le quatrième mot de la ligne. Enfin, le champ `eq_comp`, une chaine de caractère correspond à l'équation définissant le comportement du composant. Il est créé à partir du type du composant et de son nom. Par exemple, pour une inductance L_1, le champ `eq_comp` sera égal à *stateL1* (équation d'état de L_1).

Différentes fonctions permettent ensuite de calculer le nombre de nœuds du circuit, le nombre d'équations et le nombre de variables. Les équations et les variables sont ensuite créées au moyen d'un tableau de chaines de caractères. L'étape suivante consiste à écrire le système d'équations, c'est-à-dire la matrice du système. Nous allons présenter l'écriture d'une ligne correspondant à l'équation d'état d'une inductance $u_{LP} - u_{LN} - L.\dot{i_l} = 0$. Il faut d'abord balayer les lignes afin de trouver l'indice correspondant à l'équation d'état au moyen d'une identification de chaîne de caractères. Ensuite, il faut déterminer dans la liste des variables les potentiels des nœuds auxquels est connectée l'inductance. L'opération est répétée afin de trouver la dérivée du courant traversant l'inductance. Pour chacune des variables, un coefficient est écrit : *1* ou *-1* pour les potentiels, *-L* pour la dérivée du courant. Le code présenté en Annexe 1.4 illustre cette étape.

Voyons maintenant comment est écrite une équation correspondant à la loi des nœuds. Les composants connectés à un nœud ainsi que la variable (et son indice de colonne) correspondant au courant le traversant sont identifiés. Ensuite, si c'est l'interface positive du composant qui est connectée au nœud, le coefficient écrit dans la colonne correspondante est *1*, si c'est son interface négative, alors c'est *-1*. Cette étape et illustrée par le code en annexe 1.5.

Une fois la matrice du système créée, il faut réorganiser les lignes et les colonnes afin de se ramener à une structure équivalente à celle présentée en Figure 28, c'est à dire, afin d'obtenir une sous matrice X_4 de forme triangulaire supérieure. Dans un premier temps, l'algorigramme de la Figure 29 est implémenté afin de permuter les lignes. Le code en Annexe 1.6 présente cette étape. En se basant uniquement sur la sous matrice X_4, la ligne contenant le plus de termes nuls est identifiée. Cette ligne (de la matrice globale) est ensuite permutée avec la dernière ligne d'indice *n* de la matrice. Au tour de boucle suivant, la ligne considérée comme étant la dernière est celle d'indice *n-1*. Nous ne détaillerons pas l'opération consistant à permuter les colonnes, celle-ci étant assez similaire à l'opération de permutation des lignes.

La création de la matrice de relation est également une étape intéressante. A partir de la sous matrice triangulaire supérieure X_4, elle consiste à écrire les variables non nécessaires en fonction des variables nécessaires. Pour ce faire, nous nous basons sur les sous matrices X_3 et X_4. Les lignes de la matrice de relation correspondent aux variables non nécessaires, les colonnes correspondent aux variables nécessaires. La triangularisation de la sous matrice X_4 a permis de disposer les variables non nécessaires de façon à pouvoir générer la matrice de relation assez facilement. En effet, la dernière ligne de la matrice du système permet de résoudre une variable non nécessaire, de même pour la ligne supérieure et ainsi de suite. A l'issue de cette étape, les sous matrices X_3 et X_4 ne sont plus d'aucune utilité : les équations d'état constituent les lignes des sous matrices X_1 et X_2, la matrice de relation permettant d'exprimer les variables non nécessaires en fonction des variables nécessaires.

Pour pouvoir générer les matrices *A* et *B* de la représentation d'état, l'unique tâche restant à accomplir consiste à remplacer les termes non nuls de la sous matrice X_2 par leur expression en fonction des variables nécessaires, la matrice de relation facilite beaucoup cette tâche. Les matrices *C* et *D* de la représentation d'état dépendent quant à elles du choix fait par l'utilisateur des composantes du vecteur de sortie. Etant donné que ces matrices doivent permettre d'exprimer n'importe quelle variable du système en fonction des variables d'état et des entrées, c'est la matrice de relation qui sera utilisée pour les générer.

Une fois les quatre matrices *A*, *B*, *C* et *D* générées, un fichier d'en-tête (.h) est créé. Ce fichier est constitué d'un module unique contenant une classe `sca_lsf::sca_ss` paramétrée avec les matrices *A*, *B*, *C* et *D*. Ce module peut donc être utilisé dans un modèle SystemC AMS sans avoir besoin d'exécuter la méthode d'abstraction à chaque fois. Par contre, si la valeur

d'un composant de la netlist est modifiée, il faudra ré-exécuter la méthode d'abstraction afin de mettre à jour les matrices de la représentation d'état.

4.5 Exemple d'application : les circuits passifs

Pour illustrer la méthode, nous allons prendre l'exemple d'un filtre passif du quatrième ordre [Bousquet 11a] donné par la Figure 32. *Nous avertissons le lecteur que les parties en italique énumèrent les lignes et colonnes des matrices. La lecture de ces parties n'est pas essentielle pour comprendre la philosophie de la méthode présentée.*

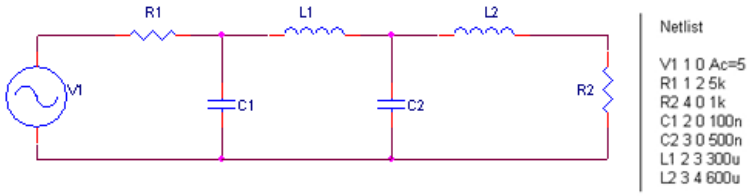


Figure 32. Filtre passif du quatrième ordre

Ce filtre comporte 2 inductances ainsi que 2 capacités, nous aurons donc 4 variables d'état. Les quatre équations d'état sont les suivantes :

$$-C_1.\dot{u_{C1}} + \dot{i}_{C1} = 0 \tag{12}$$

$$-C_2.\dot{u_{C2}} + \dot{i}_{C2} = 0 \tag{13}$$

$$-L_1.\dot{\imath_{L1}} + (u_2 - u_3) = 0 \tag{14}$$

$$-L_2.\dot{\imath_{L2}} + (u_3 - u_4) = 0 \tag{15}$$

Nous avons ensuite 4 équations dites de « topologie » :

$$-V_1 + (u_1 - u_0) = 0 \tag{16}$$

$$-i_{R1}.R_1 + (u_1 - u_2) = 0 \tag{17}$$

$$-i_{R2}.R_2 + (u_4 - u_0) = 0 \tag{18}$$

$$u_0 = 0 \tag{19}$$

Enfin, nous pouvons écrire 3 équations supplémentaires pour les lois des nœuds :

$$-i_{R1} + i_{C1} + i_{L1} = 0 \tag{20}$$

$$+i_{R2} - i_{L2} = 0 \tag{21}$$

$$+i_{C2} - i_{L1} + i_{L2} = 0 \tag{22}$$

Nous pouvons maintenant construire la matrice du système présentée par l'équation (23). *Les lignes de cette matrice correspondent respectivement et de haut en bas à : l'équation d'état de C_1, l'équation d'état de C_2, l'équation d'état de L_1, l'équation d'état de L_2, l'équation de topologie de V_1, l'équation de topologie de R_1, l'équation de topologie de R_2, l'équation définissant le potentiel du nœud de référence, la loi des nœuds appliquée au nœud N_2, la loi des nœuds appliquée au nœud N_3 et la loi des nœuds appliquée au nœud N_4. Les colonnes de la matrice correspondent respectivement et de gauche à droite aux variables : i_{R1}, i_{R2}, i_{C1}, i_{C2}, i_{L1}, i_{L2}, u_0, u_1, u_2, u_3, u_4, $\dot{u_{20}}$, $\dot{u_{30}}$, $\dot{\iota_{L1}}$, $\dot{\iota_{L2}}$ et V_1.*

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -C_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -C_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & -L_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & -L_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ -R_1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -R_2 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{23}$$

Les opérations de réorganisation donnent la matrice présentée par l'équation (24). *Les lignes de cette matrice correspondent respectivement et de haut en bas à : l'équation d'état de C_1, l'équation d'état de C_2, l'équation d'état de L_1, l'équation d'état de L_2, la loi des nœuds*

appliquée au nœud N_2, l'équation de topologie de R_1, l'équation de topologie de V_1, l'équation de topologie de R_2, la loi des nœuds appliquée au nœud N_4, la loi des nœuds appliquée au nœud N_3 et l'équation définissant le potentiel du nœud de référence. Les colonnes de la matrice correspondent respectivement et de gauche à droite aux variables : i_{L1}, i_{L2}, u_2, u_3, $\dot{u_{20}}$, $\dot{u_{30}}$, $\dot{\iota_{L1}}$, $\dot{\iota_{L2}}$, V_1, i_{C1},i_{R1}, u_1, u_4, i_{R2}, i_{C2} et u_0.

Sur cette matrice, on voit bien que la sous matrice X_2 – soit de la ligne correspondant à la loi des nœuds appliquée au nœud N_2 jusqu'à la dernière ligne et des colonnes i_{C1} jusqu'à la dernière colonne – est une matrice triangulaire supérieure.

$$\begin{pmatrix} 0 & 0 & 0 & 0 & -C_1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -C_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & -L_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -L_2 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -R_1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -R_2 & 0 & -1 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (24)$$

La matrice de relation est donc créée à partir des sous matrice X_3 et X_4. Elle est donnée par l'équation (25). *Les lignes de cette matrice correspondent respectivement et de haut en bas à : i_{C1},i_{R1}, u_1, u_4, i_{R2}, i_{C2}, soit les variables non nécessaires. Les colonnes correspondent respectivement et de gauche à droite aux variables i_{L1}, i_{L2}, u_2, u_3, $\dot{u_{20}}$, $\dot{u_{30}}$, $\dot{\iota_{L1}}$, $\dot{\iota_{L2}}$ et V_1, soit les variables nécessaires.*

$$\begin{pmatrix} -1 & 0 & {}^{-1}/_{R_1} & 0 & {}^{1}/_{R_1} \\ 0 & 0 & {}^{-1}/_{R_1} & 0 & {}^{1}/_{R_1} \\ 0 & 0 & 0 & 0 & 1 \\ 0 & R_2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \end{pmatrix} \quad (25)$$

Nous pouvons donc désormais extraire les matrices *A* et *B* de la représentation d'état afin de générer l'équation d'état du filtre passif d'ordre 4, équation (26).

$$\begin{pmatrix} \dot{u_{20}} \\ \dot{u_{30}} \\ \dot{\imath_{L1}} \\ \dot{\imath_{L2}} \end{pmatrix} = \begin{pmatrix} -1/R_1C_1 & 0 & -1/C_1 & 0 \\ 0 & 0 & 1/C_2 & -1/C_2 \\ 1/L_1 & -1/L_1 & 0 & 0 \\ 0 & 1/L_2 & 0 & -R_2/L_2 \end{pmatrix} \cdot \begin{pmatrix} u_{20} \\ u_{30} \\ i_{L1} \\ i_{L2} \end{pmatrix} + \begin{pmatrix} 1/R_1C_1 \\ 0 \\ 0 \\ 0 \end{pmatrix} . V_1 \quad (26)$$

Enfin, l'équation de sortie, équation (27) est générée à partir de la matrice de relation. Ici, nous avons défini les potentiels aux nœuds N_2 et N_4 comme étant des composantes du vecteur de sortie.

$$\begin{pmatrix} u_2 \\ u_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & R_2 \end{pmatrix} \cdot \begin{pmatrix} u_{20} \\ u_{30} \\ i_{L1} \\ i_{L2} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} . V_1 \quad (27)$$

La Figure 33 présente un comparatif entre la simulation de la représentation d'état - soit les équations (26) et (27) implémentées au moyen du MoC LSF de SystemC AMS – et celle du circuit électrique implémenté en ELN. Nous remarquons que les courbes sont parfaitement superposées. Aucune information sur le comportement en fréquence du filtre n'est perdue. D'autre part, le temps gagné en durée de simulation est de l'ordre de 10 %. Le temps de simulation est ici en réalité le temps de simulation et le temps d'élaboration. Il est mesuré comme expliqué dans le chapitre précédent. Le temps de simulation en LSF est uniquement la durée de simulation de la représentation d'état. On considère que la lecture de la netlist et la génération des matrices de la représentation d'état ne font pas partie de la simulation du modèle à proprement parler. Une fois généré, le modèle LSF peut bien sûr être réutilisé.

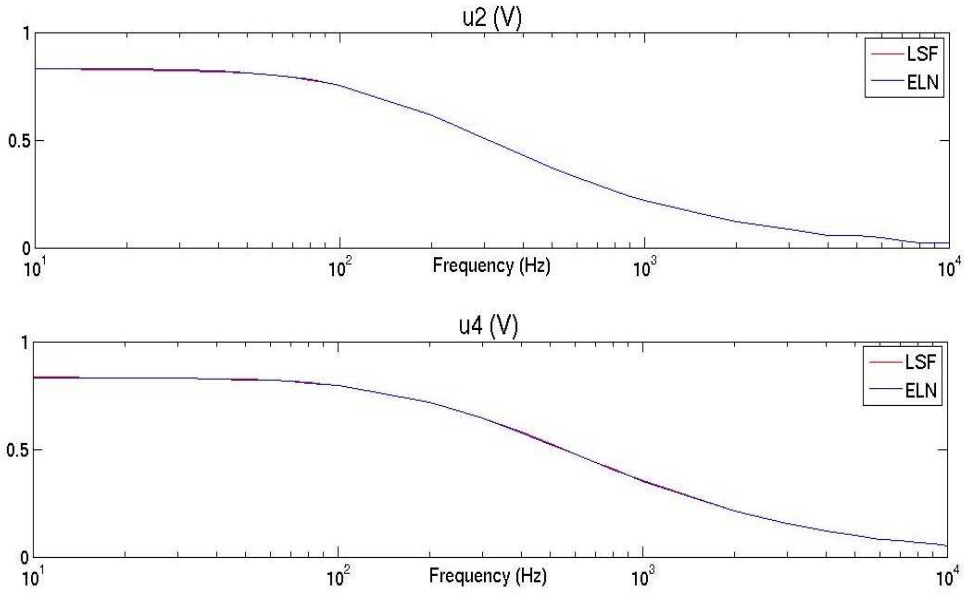


Figure 33. Comparaison LSF/ELN pour le filtre du quatrième ordre

Nous venons de présenter une méthode permettant de générer automatiquement la représentation d'état d'un circuit électrique linéaire. Le modèle créé de cette façon n'est plus conservatif contrairement au circuit électrique, seules les informations essentielles permettant d'obtenir un modèle comportemental sont gardées. La simulation de ce modèle est d'environ 10 % plus rapide pour un circuit faisant intervenir 7 modules ELN. Ceci en accord avec les résultats présentés au chapitre précédent.

Cette méthode peut être utilisée dans le cadre d'une conception top-down des parties analogiques. En effet, dans un premier temps, chaque unité fonctionnelle est modélisée par un bloc abstrait de type fonction de transfert. L'exploration architecturale consiste alors à raffiner un bloc jusqu'à trouver un circuit puis à revenir au bloc abstrait, cette procédure étant répétée pour chaque unité fonctionnelle. A ce stade, on peut se poser la question de l'utilité de la représentation d'état par rapport à une utilisation moins onéreuse de la fonction de transfert de départ. Les paragraphes suivants montreront au contraire que nous allons pouvoir enrichir le modèle abstrait avec une information de consommation, chose impossible avec une simple fonction de transfert.

4.6 Inclusion de la consommation dans les modèles de haut niveau

Le grand avantage de la représentation d'état sur la fonction de transfert est sa présentation sous une forme matricielle plus riche en information. La forme matricielle permet d'écrire non pas une équation mais un système d'équations. Le vecteur de sortie de la représentation d'état peut contenir n'importe quel signal du circuit. Il va donc être possible d'enrichir le modèle de haut niveau afin qu'il ne soit plus seulement un modèle comportemental mais un modèle intégrant des informations pouvant s'avérer intéressantes pour les ingénieurs système. Il faut cependant garder le sens de la mesure, il serait en effet tout à fait possible d'intégrer tous les signaux internes d'un circuit dans le vecteur de sortie de la représentation d'état. Cela entrainerait inévitablement un allongement non négligeable des durées de simulation, le modèle étant finalement aussi riche en information qu'une netlist. Il faudra en fait trouver le compromis idéal entre vitesse de simulation et niveau de détail du modèle. Etant donné le contexte économique et industriel, la modélisation de la consommation devient un enjeu crucial au cours du développement des systèmes intégrés. Plus l'aspect consommation est pris en compte tôt dans le flot de conception, moins il y aura de risque d'avoir de « mauvaises surprises » lors des simulations finales de validation à très bas niveau. Ainsi, il est pertinent d'enrichir les modèles de haut niveau avec les informations permettant d'estimer la consommation. La connaissance, même approximative de la consommation des différentes parties du système permet non seulement de dimensionner les alimentations nécessaires mais aussi et surtout d'identifier clairement les parties du système qui sont à optimiser. La consommation est ici prise à titre d'exemple afin d'illustrer notre méthode d'enrichissement.

4.6.1 Modélisation à haut niveau de la consommation

En ce qui concerne l'estimation de la consommation à haut niveau, un travail important a été mené et présenté dans [Lauwers 00] et [Lauwers 02]. Ce travail est essentiellement basé sur les filtres actifs. Un outil résultant de différentes méthodes a été développé : ACTIF. Dans un premier temps, une approche top-down est appliquée, l'utilisateur spécifie des informations de haut niveau telles que la fonction de transfert, la dynamique et l'amplitude maximale du signal. Ensuite, à partir d'une bibliothèque contenant les différentes topologies de filtres actifs à OTA (Operational transconductance amplifier), le filtre est synthétisé. Notre approche est assez proche de celle-ci. La consommation est ensuite estimée, puis ramenée à haut niveau. Cet outil a été développé sous MATLAB.

4.6.2 Présentation de la méthode d'inclusion de la consommation à haut niveau

Afin d'inclure l'information de consommation dans ce modèle, l'idée consiste à inclure au vecteur de sortie de la représentation d'état les courants et les tensions nécessaires à son calcul. La consommation pourra ensuite être calculée par un module TDF. En effet, le MoC LSF interdit la multiplication d'un signal par un autre. Il faudra donc convertir les signaux en TDF puis calculer la consommation. Ainsi le diagramme présenté en Figure 27 au chapitre précédent est modifié selon la Figure 34. On remarque sur celle-ci qu'une composante du vecteur de sortie de la représentation d'état est utilisée afin de calculer, il peut y en avoir plusieurs. Ces informations sont celles extraites à partir du modèle de bas niveau d'abstraction et automatiquement intégrées dans le modèle de haut niveau. Suivant cette même figure, l'autre donnée permettant de calculer la consommation est l'entrée. Par exemple, dans un circuit passif, la consommation ou la puissance absorbée est donnée par le produit de la tension d'entrée et du courant d'entrée.

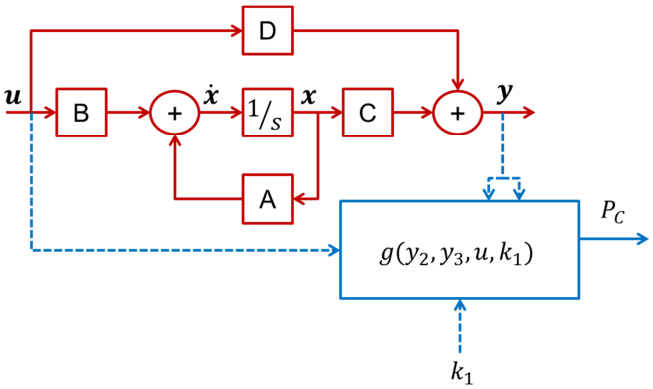


Figure 34. Diagramme schéma bloc d'un système linéaire avec calcul de la consommation

Nous allons reprendre l'exemple du filtre du quatrième ordre ayant servi de support pour illustrer la méthode d'abstraction du paragraphe 4.2.4. Ce circuit est un circuit passif : le seul apport d'énergie qu'il reçoit provient du signal d'entrée, celui que l'on souhaite filtrer. Ainsi, la puissance que le filtre absorbe est donnée par l'équation (28).

$$P_C(t) = i_{IN}(t).v_{IN}(t) \tag{28}$$

Si l'on veut créer un modèle de haut niveau du filtre passe bas basé sur la représentation d'état et qui inclut l'information relative à la consommation de puissance, c'est le seul courant d'entrée qui doit être ajouté au vecteur de sortie de la représentation d'état, la tension d'entrée n'étant pas une inconnue. Or, ce courant d'entrée i_{R1} était une variable non nécessaire. Il est donc aisé de l'extraire de la matrice de relation précédemment générée. Il peut ainsi être exprimé à partir des variables d'état et de l'entrée, il est donné par l'équation (29).

$$i_{R1} = \begin{pmatrix} -1/R_1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u_{20} \\ u_{30} \\ i_{L1} \\ i_{L2} \end{pmatrix} + \frac{V_1}{R_1} \tag{29}$$

Ainsi, la puissance instantanée absorbée par le circuit est donnée par l'équation (30).

$$P_C(t) = i_{R1}(t).V_1(t) \tag{30}$$

Il faut ensuite convertir les signaux $v_1(t)$ et $i_{R1}(t)$ du type LSF vers le type TDF au moyen des convertisseurs LSF vers TDF prédéfinis. Le code en Annexe 1.7 montre une conversion LSF vers TDF.

Le code en Annexe 1.8 montre le calcul de la puissance instantanée réalisé en TDF. C'est un produit de deux signaux qui est donc impossible à réaliser au moyen du MoC LSF.

La Figure 35 présente la tension de sortie du filtre $u_4(t)$ ainsi que son courant d'entrée $i_{R1}(t)$ et sa consommation$P_C(t)$.

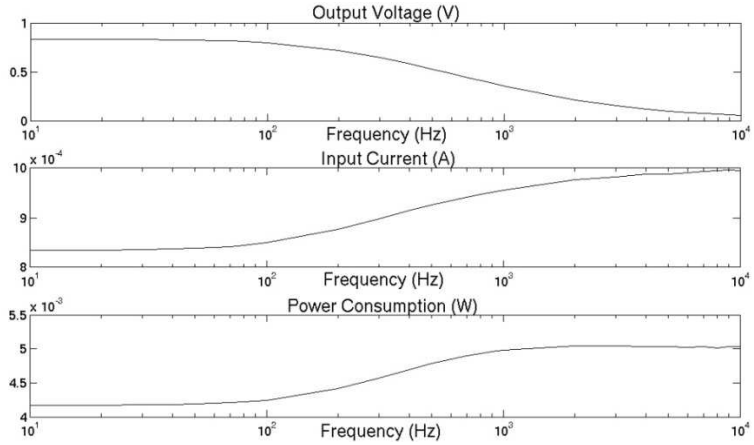
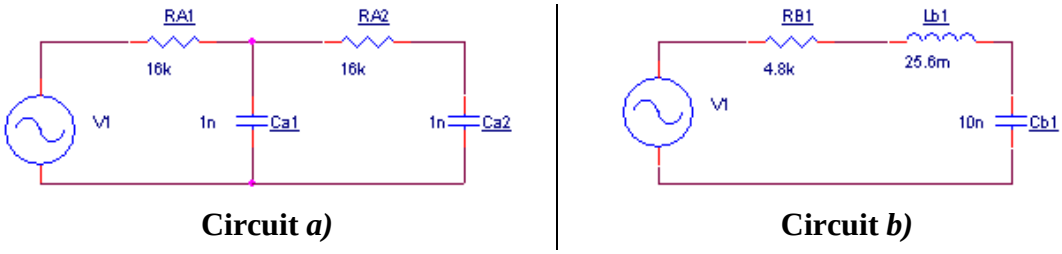


Figure 35. Tension de sortie, courant d'entrée et consommation du filtre d'ordre 4

4.6.3 Comparaison de circuits

Nous allons maintenant présenter un cas d'étude permettant de montrer l'utilité de cette méthode. Imaginons que nous devions réaliser un filtre passe bas en partant d'une fonction de transfert donnée et que ce filtre doive être réalisé uniquement à partir de composants passifs. Soit cette fonction de transfert donnée par l'équation (31). Il existe plusieurs architectures de circuit ayant un comportement correspondant à cette fonction de transfert. Imaginons maintenant que dans les spécifications, il soit précisé que la consommation de puissance de ce filtre doive être la plus faible possible autour de la fréquence de coupure. Nous pouvons aussi supposer qu'un grand nombre de simulations doivent être réalisées au niveau système et qu'il serait donc intéressant de disposer d'un modèle de haut niveau permettant de réduire les durées des simulations. Face à un tel cas, notre méthode pourra être utilisée. En effet, elle permet de générer automatiquement un modèle de haut niveau sous forme de représentation d'état et permet d'y inclure les éléments permettant de calculer la consommation. Nous allons donc comparer les deux architectures présentées en Figure 36.

$$H(s) = \frac{1}{2{,}56.10^{-10}.p^2 + 4{,}8.10^{-5}.p + 1} \tag{31}$$



Circuit *a)* **Circuit *b)***

Figure 36. Deux topologies différentes pour un même comportement

Les équations (32) et (33) présentent respectivement les matrices du système des circuits *a)* et *b)* de la Figure 36. *Dans la matrice de l'équation (32), les lignes correspondent respectivement et de haut en bas à : l'équation d'état de C_{A1}, l'équation d'état de C_{A2}, l'équation de topologie de V_1, la loi d'Ohm sur la résistance R_{A1}, la loi d'Ohm sur la résistance R_{A2}, la définition du potentiel de référence u_0, la loi des nœuds appliquée au nœud N_2 et la loi des nœuds appliquée au nœud N_3. Les colonnes de cette même matrice correspondent quant à elles respectivement et de gauche à droite aux variables : $i_{Ra1}, i_{RA2}, i_{CA1}, i_{CA2}, u_0, u_1, u_2, u_3, \dot{u_{20}}, \dot{u_{30}}$ et V_1.*

Dans la matrice de l'équation (33), les lignes correspondent respectivement et de haut en bas à : l'équation d'état de C_{B1}, l'équation d'état de L_{B1}, l'équation de topologie de V_1, la loi d'Ohm sur la résistance R_{B1}, la définition du potentiel de référence u_0, la loi des nœuds appliquée au nœud N_2 et la loi des nœuds appliquée au nœud N_3. Les colonnes de cette même

matrice correspondent quant à elles respectivement et de gauche à droite aux variables : $i_{CB1}, i_{RB1}, i_{LB1}, u_0, u_1, u_2, u_3, \dot{u_{30}}, \dot{\iota_{LB1}}$ *et* V_1.

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -C_{A1} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -C_{A2} & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ -R_{A1} & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & -R_{A2} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (32)$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & -C_{B1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & -L_{B1} & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & -R_{B1} & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (33)$$

Les équations (34) et (35) présentent respectivement les matrices réorganisées du système des circuits *a)* et *b). Les lignes de la matrice présentée par l'équation (34) correspondent respectivement et de haut en bas à : l'équation d'état de* C_{A1}*, l'équation d'état de* C_{A2}*, la loi des nœuds appliquée au nœud* N_3*, la loi des nœuds appliquée au nœud* N_2*, la loi d'Ohm sur la résistance* R_{A2}*, la loi d'Ohm sur la résistance* R_{A1}*, l'équation de topologie de* V_1 *et la définition du potentiel de référence* u_0*. Les colonnes de cette matrice correspondent respectivement et de gauche à droite aux variables :* $u_2, u_3, \dot{u_{20}}, \dot{u_{30}}, V_1, i_{CA2}, i_{CA1}, i_{RA2}, i_{RA1}, u_1$ *et* u_0*. Les lignes de la matrice présentée par l'équation (35) correspondent respectivement et de haut en bas à : l'équation d'état de* C_{B1}*, l'équation d'état de* L_{B1}*, la loi d'Ohm sur la résistance* R_{B1}*, la loi des nœuds appliquée au nœud* N_2*, la loi des nœuds appliquée au nœud* N_3*, l'équation de topologie de* V_1 *et la définition du potentiel de référence* u_0*. Les colonnes de cette même matrice correspondent quant à elles respectivement et de gauche à droite aux variables :* $i_{LB1}, u_3, \dot{\iota_{LB1}}, \dot{u_{30}}, V_1, u_2, i_{RB1}, i_{CB1}, u_1$ *et* u_0.

$$\begin{pmatrix} 0 & 0 & -C_{A1} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -C_{A2} & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & -R_{A2} & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -R_{A1} & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (34)$$

$$\begin{pmatrix} 0 & 0 & 0 & -C_{B1} & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & -L_{B1} & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & -R_{B1} & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 & -1 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (35)$$

Les équations (36) et (37) présentent les matrices de relation des *a)* et *b)* circuits. *Dans l'équation (36), les lignes de la matrice correspondent aux variables non nécessaires du circuit a), soit de haut en bas :* $i_{CA2}, i_{CA1}, i_{RA2}, i_{RA1}$ *et* u_1. *Les colonnes correspondent aux variables nécessaires, soit de gauche à droite :* u_2, u_3 *et* V_1. *Les lignes de la matrice présentée par l'équation (37) sont les variables non nécessaires du circuit b), soit de haut en bas :* u_2, i_{RB1}, i_{CB1} *et* u_1. *Les colonnes correspondent aux variables nécessaires, soit de gauche à droite :* i_{LB1}, u_3 *et* V_1.

$$\begin{pmatrix} {}^{1}/_{R_{A2}} & {}^{-1}/_{R_{A2}} & 0 \\ {}^{1}/_{R_{A1}} + {}^{1}/_{R_{A2}} & {}^{-1}/_{R_{A2}} & {}^{1}/_{R_{A1}} \\ {}^{1}/_{R_{A2}} & -{}^{1}/_{R_{A2}} & 0 \\ -{}^{1}/_{R_{A1}} & 0 & {}^{1}/_{R_{A1}} \\ 0 & 0 & 1 \end{pmatrix} \quad (36)$$

$$\begin{pmatrix} -R_{B1} & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (37)$$

Les équations (38) et (39) sont respectivement l'équation d'état et l'équation de sortie du circuit *a)*.

$$\begin{pmatrix} \dot{u_{20}} \\ \dot{u_{30}} \end{pmatrix} = \begin{pmatrix} -1/R_{A1}.C_{A1} - 1/R_{A2}.C_{A1} & 1/R_{A2}.C_{A1} \\ -1/R_{A2}.C_{A2} & -1/R_{A2}.C_{A2} \end{pmatrix} . \begin{pmatrix} u_{20} \\ u_{30} \end{pmatrix} + \begin{pmatrix} 1/R_{A1}.C_{A1} \\ 0 \end{pmatrix} . V_1 \quad (38)$$

$$\begin{pmatrix} u_3 \\ i_{RA1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1/R_{A1} & 0 \end{pmatrix} . \begin{pmatrix} u_{20} \\ u_{30} \end{pmatrix} + \begin{pmatrix} 0 \\ 1/R_{A1} \end{pmatrix} . V_1 \quad (39)$$

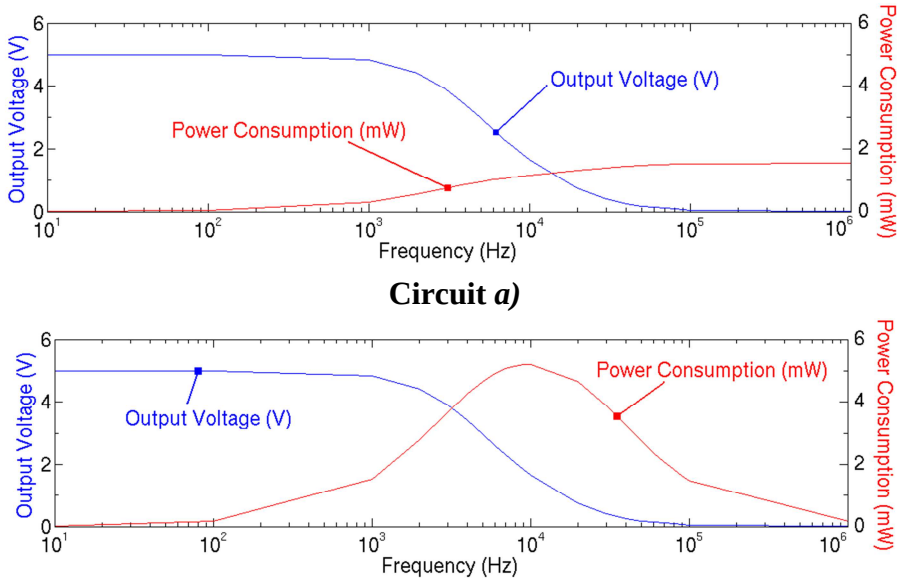
Les équations (40) et (41) sont respectivement l'équation d'état et l'équation de sortie du circuit *b)*.

$$\begin{pmatrix} \dot{\imath_{LB1}} \\ \dot{u_{30}} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} . \begin{pmatrix} i_{LB1} \\ u_{30} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} . V_1 \quad (40)$$

$$\begin{pmatrix} u_3 \\ i_{RB1} \end{pmatrix} = \begin{pmatrix} -R_{B1}/L_{B1} & -1/L_{B1} \\ 1/C_{B1} & 0 \end{pmatrix} . \begin{pmatrix} i_{LB1} \\ u_{30} \end{pmatrix} + \begin{pmatrix} 1/L_{B1} \\ 0 \end{pmatrix} . V_1 \quad (41)$$

Ces deux représentations d'état incluent l'information permettant de calculer la puissance consommée dans chaque cas. En effet, la deuxième composante du vecteur de sortie est, dans les deux cas, le courant d'entrée du circuit. Sur un filtre passif, la puissance est donnée par le produit du courant d'entrée et de la tension d'entrée.

Enfin, la Figure 37 présente le diagramme de gain des deux circuits. Nous remarquons que le circuit *b)* consomme beaucoup plus de puissance pour une fréquence du signal d'entrée proche de la fréquence de coupure. En effet, ce filtre est de type RLC, le comportement est passe bas car la sortie est la tension aux bornes du condensateur mais le phénomène de résonnance à la fréquence de coupure entraîne la surintensité et donc une consommation plus importante.



Circuit *a)*

Circuit *b)*

Figure 37. Comparaison tensions de sortie et puissances

4.6.4 Les circuits actifs linéaires

Il est possible d'étendre cette méthode aux circuits actifs linéaires bâtis autour d'amplificateurs opérationnels. En effet, en considérant que ces composants fonctionnent de façon idéale et en régime linéaire, deux équations définissent leur comportement. La première permet d'exprimer l'égalité des potentiels en entrée, cette équation apparaît dans le Tableau 10. La seconde, qui ne nécessite pas d'être ajoutée au système d'équation précise que les courants sur les entrées de l'amplificateur opérationnels sont nuls. Il n'est pas obligatoire d'ajouter cette seconde équation à la matrice, ce qui ne serait pas le cas si les courants d'entrées n'étaient pas considérés comme étant nuls. En effet, le fait de ne pas ajouter l'équation fait que ces courants sont automatiquement considérés comme étant nuls. Ainsi, la matrice est allégée d'une équation. Comme exemple, nous allons traiter le cas d'un filtre à variable d'état. Cette architecture de filtre est intéressante car elle dispose des trois types de sorties : passe haut, passe bas et passe bande. La Figure 38 présente un filtre à variable d'état d'ordre 2. Au nœud 3, nous retrouvons la sortie passe haut. Au nœud 5, nous retrouvons la sortie passe bande. Enfin, au nœud 7, nous retrouvons la sortie passe bas.

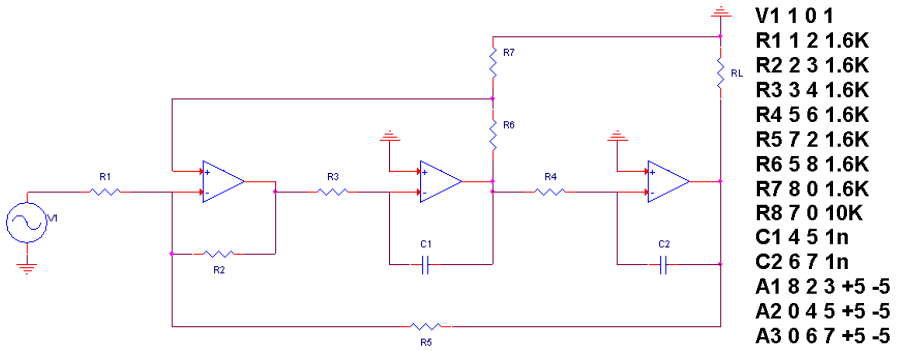


Figure 38. Filtre à variables d'état du second ordre et sa netlist

L'application de la méthode décrite précédemment nous permet de générer la matrice du circuit, que nous retrouvons dans l'équation (42). *Les lignes de cette matrice correspondent respectivement et de haut en bas à : l'équation d'état de* C_1, *l'équation d'état de* C_2, *l'équation de topologie de* V_1, *la loi d'Ohm sur la résistance* R_1, *la loi d'Ohm sur la résistance* R_2, *la loi d'Ohm sur la résistance* R_3, *la loi d'Ohm sur la résistance* R_4, *la loi d'Ohm sur la résistance* R_5, *la loi d'Ohm sur la résistance* R_6, *la loi d'Ohm sur la résistance* R_7, *la loi d'Ohm sur la résistance* R_L, *la définition de la différence de potentiel* u_{45}, *la définition de la différence de potentiel* u_{67}, *l'équation définissant le comportement linéaire de* A_3, *l'équation définissant le comportement linéaire de* A_2, *l'équation définissant le comportement linéaire de* A_1, *la définition du potentiel de référence* u_0, *la loi des nœuds appliquée au nœud* N_2, *la loi des nœuds appliquée au nœud* N_4, *la loi des nœuds appliquée au nœud* N_8 *et la loi des nœuds appliquée au nœuds* N_6. *Les colonnes de cette matrice correspondent respectivement et de gauche à droite aux variables :* $\dot{u_{45}}, \dot{u_{67}}, u_{45}, u_{67}, V_1, u_0, u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, i_{R1}, i_{R2}, i_{R3}, i_{R4}, i_{R5}, i_{R6}, i_{R7}, i_{RL}, i_{C1}$ *et* i_{C2}.

Après la réorganisation de la matrice, nous obtenons la nouvelle forme de la sous matrice X_4, X_4 apparaît dans l'équation (43).

$$\begin{pmatrix}
-C_1 &&&&& &&&&& &&&&& &&&&& && 1 & \\
& -C_2 &&&&& &&&&& &&&&& &&&&& && 1 \\
&&&& -1 & -1 & 1 &&&&& &&&&& &&&&& && \\
&&&&&& 1 & -1 &&&&& && -R_1 &&&&& &&&& \\
&&&&&&& 1 & -1 &&&&& && -R_2 &&&&& &&& \\
&&&&& &&& 1 & -1 &&&&& && -R_3 &&&&& && \\
&&&&& &&&&& 1 & -1 &&&&& & -R_4 &&&&& & \\
&&&&& && 1 &&&&& & -1 &&&&& & -R_5 &&&&& \\
&&&&& &&&&& 1 &&& -1 &&&&& & -R_6 &&&& \\
&&&&& 1 &&&&& &&& 1 &&&&& && -R_7 &&& \\
&&&&& -1 &&&&& && 1 &&&&& &&&& -R_L && \\
&& -1 &&&&& && 1 & -1 &&&&& &&&&& &&& \\
&&& -1 &&&&& &&& 1 & -1 &&&&& &&&&& & \\
&&&&& 1 &&&&& & -1 &&&&& &&&&& && \\
&&&&& 1 &&&& -1 &&&&& &&&&& &&&& \\
&&&&& && -1 &&&&& & 1 &&&&& &&&&& \\
&&&&& -1 &&&&& &&&&& &&&&& &&& \\
&&&&& &&&&& &&&& -1 & 1 &&& 1 &&&&& \\
&&&&& &&&&& &&&&& & -1 &&&&& & 1 & \\
&&&&& &&&&& &&&&& &&&& -1 & 1 &&& \\
&&&&& &&&&& &&&&& && 1 &&&&& & 1
\end{pmatrix} \quad (42)$$

$$\begin{pmatrix}
1 & -1 & & & & & & & & & & & & & & & & \\
 & -R_3 & 1 & & & & & & & & & & & & -1 & & & \\
 & & -1 & -R_2 & & & & & & & -1 & & & & & & & \\
 & & & 1 & -1 & & & -1 & & & & & & & & & & \\
 & & & & -R_1 & 1 & & & & & -1 & & & & & & & \\
 & & & & & 1 & & & & & & & & & & & -1 & \\
 & & & & & & 1 & -1 & & & & & & & & & & \\
 & & & & & & & -R_4 & & & & & & 1 & & -1 & & \\
 & & & & & & & & -R_5 & & & -1 & & 1 & & & & \\
 & & & & & & & & & -1 & & & {}^{1}/_{R_6+R_7} & & & & & \\
 & & & & & & & & & & -1 & & {}^{1}/_{R_6+R_7} & & & & & \\
 & & & & & & & & & & & -R_L & & 1 & & & -1 & \\
 & & & & & & & & & & & -1 & 1 & & & & & \\
 & & & & & & & & & & & & -1 & {}^{R_7}/_{R_6+R_7} & & & & \\
 & & & & & & & & & & & & & -1 & & 1 & & \\
 & & & & & & & & & & & & & & -1 & & 1 & \\
 & & & & & & & & & & & & & & & -1 & & 1 \\
 & & & & & & & & & & & & & & & & -1 & 1 \\
 & & & & & & & & & & & & & & & & & 1
\end{pmatrix} \tag{43}$$

Les lignes de la sous matrice X_4 correspondent respectivement et de haut en bas à : la loi des nœuds appliquée au nœud N_4, la loi d'Ohm sur la résistance R_3, la loi d'Ohm sur la résistance R_2, la loi des nœuds appliquée au nœud N_2, la loi d'Ohm sur la résistance R_1, l'équation de topologie de V_1, la loi des nœuds appliquée au nœuds N_6, la loi d'Ohm sur la résistance R_4, la loi d'Ohm sur la résistance R_5, la loi d'Ohm sur la résistance R_6, la loi d'Ohm sur la résistance R_7, la loi d'Ohm sur la résistance R_L, l'équation définissant le comportement linéaire de A_1, la définition de la différence de potentiel u_{45}, la loi des nœuds appliquée au nœud N_8, la définition de la différence de potentiel u_{67}, l'équation définissant le comportement linéaire de A_2, l'équation définissant le comportement linéaire de A_3 et la définition du potentiel de référence u_0. La matrice globale du système comporte évidemment deux lignes supplémentaires : l'équation d'état de C_1 et l'équation d'état de C_2 qui sont les deux lignes supérieures. Les colonnes de la sous matrice X_4 correspondent respectivement et de gauche à droite aux variables : $i_{C1}, i_{R3}, u_3, i_{R2}, i_{R1}, u_1, i_{C2}, i_{R4}, i_{R5}, i_{R6}, i_{R7}, i_{RL}, u_2, u_8, u_5, u_7, u_4, u_6$ et u_0. La matrice globale du système comporte 5 colonnes supplémentaires ; les variables nécessaires que sont : $\dot{u_{45}}, \dot{u_{67}}, u_{45}, u_{67}$ et V_1qui sont disposées sur les colonnes de gauche.

Finalement, les deux équations de la représentation d'état sont extraites. L'équation (44) correspond à l'équation d'état et l'équation (45) correspond à l'équation de sortie. Ici, nous avons considéré que le nœud 7 était la sortie du système que nous voulions analyser. La Figure 39 montre le résultat de simulation. C'est la tension entre le nœud 7 et le nœud de référence qui a été tracée.

$$\begin{pmatrix} \dot{u_{45}} \\ \dot{u_{67}} \end{pmatrix} = \begin{pmatrix} -9{,}37.10^5 & 6{,}25.10^5 \\ -6{,}25.10^5 & 0 \end{pmatrix} . \begin{pmatrix} u_{45} \\ u_{67} \end{pmatrix} + \begin{pmatrix} 6{,}25.10^5 \\ 0 \end{pmatrix} . V_{IN} \tag{44}$$

$$u_{70} = (0 \quad -1) . \begin{pmatrix} u_{45} \\ u_{67} \end{pmatrix} \tag{45}$$

Les résultats donnés par la représentation ont été comparés avec un circuit ELN. Le code correspondant est donné en Annexe 1.9. On remarque qu'une résistance de faible valeur a été placée entre le nœud N_7 et la sortie du module. Il est en effet impossible qu'un nœud ELN soit directement connecté à un port ELN.

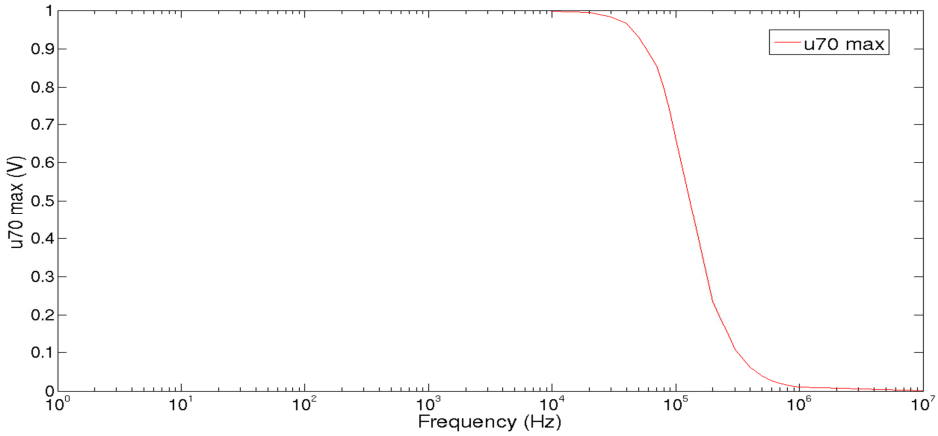


Figure 39. Tension de sortie du filtre à variables d'état

Si l'on désire maintenant enrichir la représentation d'état avec les informations permettant de calculer la consommation, il va nous falloir enrichir le vecteur de sortie de la même façon que dans la partie précédente traitant des circuits passifs. Néanmoins, à la différence d'un circuit passif, une partie seulement de la puissance consommée provient de l'entrée. L'autre partie (non négligéable) provient des alimentations des amplificateurs opérationnels. D'autre part, il ne faut pas négliger la consommation propre des AOP. Cette consommation est exprimée par l'équation (46). Le courant i_Q est le courant de repos des AOP. Cette puissance peut être appelée puissance statique.

$$P_Q = 3.i_Q.(V_{DD} - V_{SS}) \tag{46}$$

L'autre partie de la puissance consommée, la puissance dynamique peut être calculée à partir de certains courants du circuit. En effet, nous pouvons dire que le courant absorbé par le circuit (à l'exception des AOP) est la somme des courants dans les branches connectées à la masse. Dans notre filtre à variables d'état, ces courants sont i_{RL} et i_{R8}. Nous générons une nouvelle équation (47) de sortie au sens de la représentation d'état qui inclue ces deux courants.

$$\begin{pmatrix} u_{70} \\ i_{RL} \\ i_{R8} \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ -3{,}13.10^{-4} & 0 \\ 0 & -1.10^{-5} \end{pmatrix} . \begin{pmatrix} u_{45} \\ u_{67} \end{pmatrix} \tag{47}$$

Ainsi, la partie dynamique de la consommation est donnée par le produit de ces courants et de la tension d'alimentation. Nous considérons que la tension d'alimentation est ici la tension d'alimentation des amplificateurs opérationnels. Soit la partie dynamique de la consommation du filtre à variable d'état donnée par l'équation (48).

$$P_D = (i_{R7} + i_{RL}).(V_{DD} - V_{SS}) \tag{48}$$

Enfin, la consommation globale du circuit est donnée par la somme de la consommation dite statique et de la consommation dite dynamique en équation (49).

$$P_C = (V_{DD} - V_{SS}).[3.i_Q + (i_{R7} + i_{RL})] \quad (49)$$

Les simulations ont été réalisées à partir d'un signal d'entrée sinusoïdal de 1 Volt d'amplitude. La valeur du courant de repos des amplificateurs opérationnels est celle d'un circuit TL081 soit 1,4 mA. La Figure 40 montre l'évolution de la consommation globale du circuit P_C. 10^5 Hz est la fréquence de résonnance de la tension de sortie du second amplificateur opérationnel (u_{50}). A cette fréquence, u_{50} atteint sa valeur maximum (1 V) et u_{70} atteint 0,707 V (valeur à la fréquence de coupure à Gain maximum – 3 dB). Le résultat de cette combinaison est un pic de consommation à cette fréquence puisque les deux courants i_{R7} et i_{RL} dépendent respectivement des valeurs des tensions u_{50} et u_{70}.

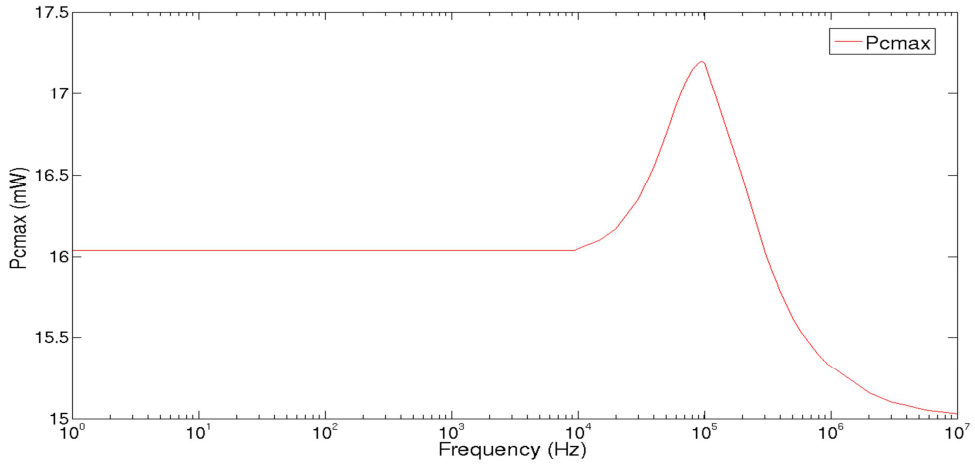


Figure 40. Consommation de puissance du filtre à variables d'état

4.7 Conclusion du chapitre

Au cours de ce chapitre, nous avons vu que les modèles de haut niveau créés peuvent être enrichi avec des informations permettant de calculer la consommation de puissance. Ces modèles enrichis demeurent plus légers que des modèles conservatifs. Il est important de noter que si tous les signaux du circuit sont ajoutés au vecteur de sortie de la représentation d'état, le modèle contient alors autant d'informations que le circuit. Son temps de simulation s'accroit alors considérablement. C'est pour cela qu'il faut savoir garder le sens de la mesure : il ne faut ramener à haut niveau que les informations pertinentes. L'objectif n'est donc pas seulement de traduire un circuit électrique en équations pour créer un modèle mathématique. Il faut sélectionner les informations essentielles qui, ramenées à haut niveau permettent de confronter le modèle avec les spécifications. Ce travail a fait l'objet de plusieurs publications [Bousquet 11a][Bousquet 11b][Bousquet 12].

5. Modélisation des systèmes électromécaniques par circuits équivalents

Ce chapitre est consacré à la modélisation des systèmes multiphysique, plus particulièrement des systèmes électromécaniques. Ces systèmes permettent l'interaction entre les différents domaines de la physique. Comme exemples de ces systèmes, nous citerons :

- la turbine hydroélectrique (hydraulique, mécanique, électricité).
- le thermocouple (thermique, mécanique, électricité).

Les systèmes embarqués d'aujourd'hui sont évidemment hétérogènes, ils intègrent une multitude de capteurs et d'actionneurs tels qu'on en trouve sur un véhicule automobile, par exemple. Après avoir proposé une méthode pour la modélisation des circuits électriques linéaires à haut niveau, c'est tout naturellement que nous nous sommes penchés sur les systèmes électromécaniques. Il apparaît la nécessité de déterminer les outils et les méthodes permettant de modéliser au mieux les phénomènes propres à chacun des domaines de la physique mais aussi les interactions se produisant entre ces domaines.

A ce jour, les deux principales méthodes existantes qui permettent de produire le modèle d'un système multiphysique sont d'une part la méthode du circuit équivalent électrique et d'autre part la méthode bond graph. Elles permettent toutes deux de générer un modèle unique, simulable au moyen d'un outil unique, soit un simulateur électrique soit un simulateur dédié aux bond graphs. Ce chapitre est consacré aux circuits équivalents électriques, le chapitre 6 sera consacré aux bond graphs.

Nous allons commencer par aborder le principe de l'analogie entre les domaines physiques, concept sur lequel repose le circuit électrique équivalent et les bond graphs. Nous ne présenterons ici que l'analogie directe, sur laquelle est basée la méthode bond graph. Il existe aussi l'analogie indirecte.

5.1 Analogie électromécanique directe

Prenons l'exemple d'une des lois parmi les plus connues de la physique : la deuxième loi de Newton ou principe fondamental de la dynamique (PFD). Cette loi, donnée par l'équation (50) stipule que l'accélération subie par un objet de masse *m* est proportionnelle à la somme des forces qu'il subit et inversement proportionnel à sa masse *m*.

$$\sum_i \vec{F_i} = m.\vec{a} \tag{50}$$

Si nous étudions un corps qui n'est soumis qu'à la force de gravité, alors nous pouvons écrire l'équation (51), avec *v* la vitesse de l'objet.

$$F_m = m.\frac{dv_m}{dt} \tag{51}$$

Prenons maintenant l'exemple d'une inductance. L'équation (52) montre la loi régissant le comportement d'une inductance.

$$u_L = L.\frac{di_L}{dt} \tag{52}$$

En comparant les équations (51) et (52), nous voyons que :

- la tension joue le même rôle que la force.
- Le courant joue le même rôle que la vitesse.

Ainsi, nous pouvons dire qu'une inductance joue le même rôle dans le domaine de l'électricité qu'une masse dans le domaine mécanique. Ces observations sont valables pour la résistance électrique et la résistance visqueuse (amortisseur) de même que pour la capacité et le ressort. Le Tableau 11 montre les équivalences entre ces composants ainsi que les lois régissant leurs comportements.

Tableau 11. Equivalence entre les composants linéaires électriques et mécaniques

Domaine mécanique		**Domaine électrique**	
Résistance visqueuse	$F_b = -b.u_b$	Résistance	$v_R = R.i_R$
Ressort	$F_k = -k.x_k$	Capacitance	$i_C = C\frac{du_C}{dt}$
Masse	$F_m = m.\frac{dv_m}{dt}$	Inductance	$u_L = L.\frac{di_L}{dt}$

Nous pouvons donc généraliser : un système physique est défini en termes d'effort, noté *e* et de flux, noté *f*. Dans le domaine mécanique, la force est considérée comme un effort et la vitesse comme un flux. Dans le domaine électrique, c'est la tension qui est considérée comme étant un effort et le courant est considéré comme un flux. L'effort et le flux sont des variables de puissance, on les appelle, indépendamment du domaine physique, variables généralisées d'effort et de flux. La puissance est égale au produit de l'effort et du flux, comme cela apparat dans l'équation (53).

$$P(t) = e(t).f(t) \tag{53}$$

L'énergie est donnée par l'intégration de la puissance par rapport au temps, comme cela apparaît dans l'équation (54).

$$E(t) = \int_0^T P(t).dt + E(0) \tag{54}$$

Deux variables d'énergie sont aussi définies : le moment généralisé noté *p(t)* et le déplacement généralisé *q(t)*, données respectivement par l'équation (55) et l'équation (56).

$$p(t) = \int_0^T e(t).dt + p(0) \tag{55}$$

$$q(t) = \int_0^T f(t).dt + q(0) \tag{56}$$

Le Tableau 12 fait apparaître les variables de puissance et d'énergie suivant le domaine physique. Les équivalences entre la mécanique de rotation et la mécanique de translation sont évidentes.

Tableau 12. Variables de puissance et d'effort pour les domaines mécanique et électrique

	Mécanique Translation	**Mécanique Rotation**	**Electricité**
Effort ***e***	Force *F*	Couple *τ*	Tension *v*
Flux ***f***	Vitesse *u*	Vitesse angulaire *ω*	Courant *i*
Moment **p**	Moment P	Moment angulaire h	Flux magnétique Φ
Déplacement ***q***	Déplacement *x*	Angle *θ*	Charge *q*

Les simulateurs électriques de type SPICE sont très performants, ils sont de plus très simples d'utilisation. En se basant sur les observations précédentes et les analogies existantes entre les différents domaines physiques, l'idée de modéliser les parties mécaniques d'un système par leur équivalent électrique apparaît clairement.

5.2 La méthode du circuit électrique équivalent

Lorsque l'on veut produire le circuit équivalent électrique d'un système, nous ramenons toutes les parties à leur équivalent électrique en utilisant les analogies. Nous entendons par là que le système physique dans son intégralité sera modélisé par un circuit électrique. L'avantage de cette méthode est qu'il existe nombre de simulateurs électriques efficaces, de plus, la partie électrique n'est pas concernée par le travail de recherche d'équivalence : elle s'intégrera telle quelle dans le circuit équivalent, il faudra simplement déterminer quels sont les éléments adéquats à placer à ses frontières pour gérer les interactions avec les autres domaines. L'inconvénient est que les simulateurs électriques, aussi performants soient-ils restent relativement lents par rapport à des simulateurs se basant sur des modèles de haut niveau. Pour remédier à ce problème, une fois le circuit équivalent électrique connu, nous pourrons appliquer la méthode présentée dans le chapitre 4 afin de créer un modèle de haut niveau. Les systèmes électromécaniques et leurs circuits électriques équivalents pris à titre d'exemples dans ce chapitre sont présentés dans les publications de référence [Tilmans 96] et [Tilmans 97].

Pour déterminer l'équivalent électrique d'une partie mécanique, il faut d'abord déterminer son dual. En effet, deux éléments mécaniques associés en série sont soumis à la même force donc au même effort alors que deux éléments électriques associés en série seront eux soumis au même courant, soit au même flux. Inversement, deux éléments mécaniques associés en parallèle évolueront à la même vitesse (flux), alors que deux éléments électriques associés en parallèle seront soumis à la même tension (effort). La seconde étape consiste à remplacer les éléments des domaines autres qu'électrique par leur équivalent électrique en respectant les Unités du Système International. Par exemple, une masse de 100 grammes aura pour équivalent électrique une inductance de 0,1 Henry. Enfin, les interactions entre les domaines comme un moteur électrique (interaction électromécanique) seront modélisées au moyen d'un transformateur ou d'un gyrateur. Au primaire du transformateur, on retrouvera un domaine physique, par exemple le domaine électrique et au secondaire, le domaine mécanique. Un rapport de transformation ayant une valeur égale à 1 signifie qu'une tension de 1 Volt au primaire donnera une force de 1 Newton au secondaire. Un gyrateur effectuant la même fonction électromécanique fera lui correspondre 1 Volt au primaire à une vitesse de 1 rad/s (dans le cas d'un moteur électrique) au secondaire. Ces composants serviront aussi à modéliser des rapports de transformation propres à un domaine. Par exemple, un système d'engrenages ou un système poulies-courroie sera modélisé par un transformateur.

Afin d'établir le circuit électrique équivalent d'un système, on distingue deux cadres: les systèmes à paramètres localisés d'une part et les systèmes à paramètres distribués d'autre part. Mais voyons tout d'abord les éléments qui vont nous permettre de gérer l'interface entre le domaine mécanique et le domaine électrique : les transducteurs.

5.3 Les transducteurs

Les transducteurs sont les éléments permettant de convertir un signal d'un domaine physique à un autre domaine physique. Prenons l'exemple d'un microphone dynamique, système multiphysique puisque faisant intervenir les domaines acoustique, mécanique et électrique. Dans un tel système, une bobine est fixée à une membrane et placée autour d'un aimant permanent. L'effet (la variation) de la pression acoustique entraine un déplacement de la membrane et donc de la bobine. La bobine mobile étant placée dans un champ magnétique, un courant électrique se forme. Le dispositif bobine mobile – aimant permanent constitue un transducteur électrodynamique. Le haut-parleur électrodynamique permet, lui, de convertir un signal électrique en signal sonore. Le dispositif est identique. Ici, un courant électrique traverse une bobine qui est placée dans un champ magnétique généré par un aimant permanent. Ceci entraine un déplacement de la bobine, et donc de la membrane à laquelle elle est fixée. Le déplacement de la membrane entraine un déplacement d'air, ce qui produit un son. La Figure 41 présente quatre types de transducteurs électromécaniques : le transducteur électrostatique transversal, le transducteur électrostatique parallèle, le transducteur électromagnétique et le transducteur électrodynamique.

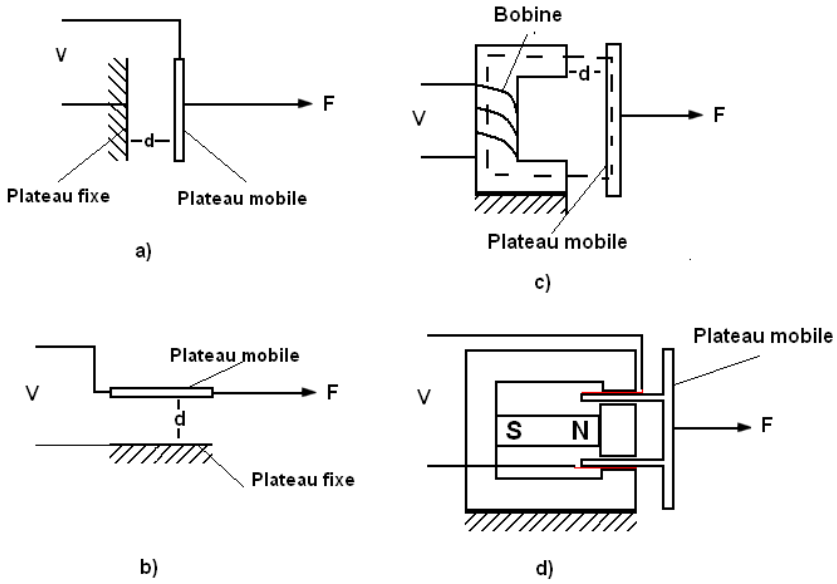


Figure 41. Différents types de transducteurs électromécaniques

a) Transducteur électrostatique transverse. b) Transducteur électrostatique parallèle. c) Transducteur électromagnétique. d) Transducteur électrodynamique.

Dans [Tilmans 96], nous trouvons différents circuits électriques équivalents pour chacun de ces transducteurs. La Figure 42 présente le modèle que nous avons choisi dans chaque cas afin de construire une bibliothèque SystemC AMS de transducteurs. Cependant, une fois que l'on dispose de la topologie du circuit équivalent, il faut déterminer les valeurs des éléments du circuit. La partie du circuit équivalent correspondant à la partie électrique demeure inchangée, que ce soit au niveau de la topologie ou des valeurs des composants. L'interface, modélisée par un transformateur ou un gyrateur permet de définir un rapport de transduction. Si l'on considère des transducteurs électriques vers mécanique alors ce rapport s'exprimera en Newtons/Ampère pour le transducteur électromagnétique et le transducteur électrodynamique et en Newtons/Volt pour les transducteurs électrostatiques. Le Tableau 13 donne les rapports de transductions suivant le type du transducteur. Ce rapport est fonction des paramètres qui peuvent être des paramètres physiques du système ou des grandeurs absolues comme la permittivité du vide ou encore des valeurs de polarisation. Les définitions des paramètres et leurs unités sont données dans le Tableau 14.

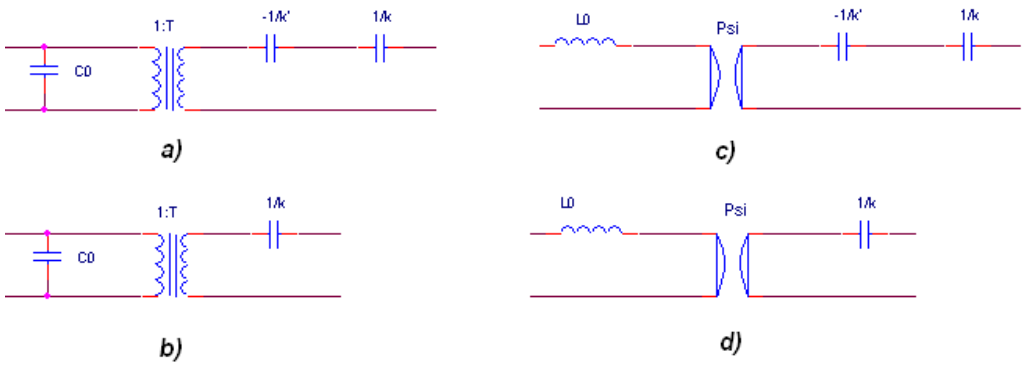


Figure 42. Circuits équivalents des transducteurs électromécaniques couplés à un ressort

a) Circuit équivalent du transducteur électrostatique transverse. b) Circuit équivalent du transducteur électrostatique parallèle. c) Circuit équivalent du transducteur électromagnétique. d) Circuit équivalent du transducteur électrodynamique.

Tableau 13. Rapport de transduction suivant le type de transducteur

Type de transducteur	Rapport de transduction	Unité
Electrostatique transverse	$\Gamma = \frac{\varepsilon_0 . A_e . V_0}{(d + x_0)^2}$	$N.V^{-1}$
Electrostatique parallèle	$\Gamma = \frac{\varepsilon_0 . h . V_0}{d}$	$N.V^{-1}$
Electromagnétique	$\Psi = \frac{N^2 . \mu_0 . A_e . I_0}{(d + x_0)^2}$	$N.A^{-1}$
Electrodynamique	$\Psi = \beta_0 . l$	$N.A^{-1}$

Tableau 14. Paramètres des transducteurs

Paramètre	Définition	Unité
ε_0	Permittivité du vide	$F.m^{-1}$
A_e	Surface active de l'électrode	m^2
V_0	Tension de polarisation	V
d	Ecart au repos	m
x_0	Position de polarisation	m
h	Epaisseur de la poutre	m
N	Nombre de spires de la bobine	-
μ_0	Perméabilité du vide	$T.m.A^{-1}$
I_0	Courant de polarisation	A
β_0	Champ magnétique de polarisation	T
l	Longueur active de la bobine	m

Les exemples de la Figure 42 font apparaître un ressort de constante de raideur *k* couplé avec le transducteur. Alors que l'analogie est directe entre un amortisseur et une résistance électrique et entre une masse et une inductance, ce n'est pas la raideur mais son inverse qui est analogue à la capacité. Pour définir cette grandeur, c'est souvent le terme anglais que l'on utilise : compliance (que l'on pourrait traduire par flexibilité ou souplesse). Les valeurs des éléments constituant les circuits équivalents de la Figure 42 sont données dans le Tableau 15.

Tableau 15. Valeurs des éléments constituant les transducteurs de la Figure 42

Type de transducteur	**Elément**		
Electrostatique transverse	$C_0 = \frac{\varepsilon_0 . A_e}{d + x_0}$	$k' = \frac{\Gamma^2}{C_0}$	$k^* = k - k'$
Electrostatique parallèle	$C_0 = \frac{\varepsilon_0 . (l_0 - x_0) h}{d}$		
Electromagnétique	$L_0 = \frac{N^2 . \mu_0 . A_e}{d + x_0}$	$k' = \frac{\Psi^2}{L_0}$	$k^* = k - k'$
Electrodynamique	$L_0 = L$ (valeur de la bobine mobile)		

Nous avons constitué une bibliothèque qui dispose de huit modèles : un transducteur mécanique vers électrique et un transducteur électrique vers mécanique pour chacun des quatre types de transducteur. Le code en Annexe 2.1 montre l'exemple du transducteur électrodynamique électrique/mécanique, celui que l'on retrouve dans le cas d'un haut-parleur électrodynamique par exemple.

5.4 Les systèmes à paramètres localisés

L'analyse d'un système à paramètres localisés est la plus simple. Si l'on fait l'hypothèse qu'un système est à paramètres localisés, on considère que les propriétés physiques telles que la masse ou la raideur sont concentrés sur un seul élément physique. Ainsi, on considère ces éléments comme étant parfaits : un élément représentant une masse est parfaitement rigide et un ressort n'a pas de masse. Ce cadre est similaire à la méthode consistant à considérer sur un circuit électrique qu'une inductance n'a aucun effet capacitif ou résistif. Cette hypothèse est vérifiée tant que la longueur d'onde du signal est plus grande que toutes les dimensions du système. Nous allons appliquer la méthode du circuit équivalent à deux exemples, un capteur de vibrations dans un premier temps puis un filtre électromécanique.

5.4.1 Capteur de vibrations

La Figure 43 montre le capteur de vibrations [Tilmans 96] que nous nous proposons d'étudier. Ce dispositif est basé sur un transducteur électrodynamique. La bobine mobile est fixée à une

masse, elle-même suspendue au socle vibrant par l'intermédiaire d'un amortisseur et d'un ressort. Ce dispositif a un seul degré de liberté. La Figure 44 présente le circuit électrique équivalent du capteur de vibrations. La partie de droite, dont la référence est *gnde* est la partie électrique. La partie de gauche dont la référence est *gndm* représente la partie mécanique. La partie encadrée en bleu correspond au transducteur électrodynamique. La partie électrique demeure inchangée par rapport à la Figure 43, à ceci près que nous avons ajouté une résistance R_S afin de modéliser la résistance série de la bobine. Le transducteur électrodynamique est modélisé par l'association de l'inductance (bobine mobile) et du gyrateur de facteur de transduction Ψ. La grandeur mécanique qui est convertie en signal électrique est la vitesse relative de la masse par rapport au socle. Ainsi, la source de courant i_{IN}, qui modélise la source de vibrations et la masse modélisée par une inductance L_1 en mouvement sont associés en parallèle. Le ressort est modélisé par une capacité C_1 et l'amortisseur est modélisé par une résistance R_1. Ces deux éléments sont soumis à la même force, ils sont donc associés en série (soit traversé par le même courant) dans le circuit équivalent électrique. La fonction de transfert v_e/u_{in}, déterminée à partir du circuit électrique équivalent est donnée par l'équation (57) où $Q = \frac{L_1.\omega_0}{c}$ et $\omega_0 = \sqrt{\frac{k}{L_1}}$.

$$\frac{v_e}{u_{in}} = \frac{i.\omega.L_1.\Psi.R_L}{\Psi^2 + \frac{k}{i.\omega}\left(1 + \frac{1}{Q}\left(\frac{i.\omega}{\omega_0}\right) + \left(\frac{i.\omega}{\omega_0}\right)^2\right)(R_L + R_S + i.\omega.L_0)} \tag{57}$$

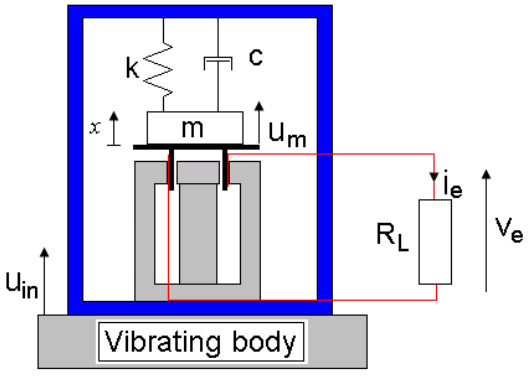


Figure 43. Capteur de vibrations

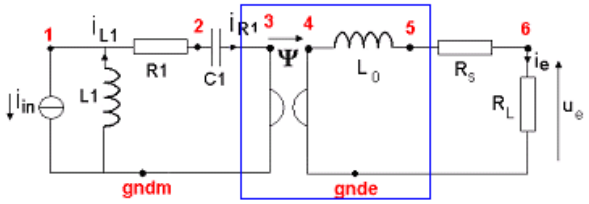


Figure 44. Circuit électrique équivalent du capteur de vibrations

A partir de ce circuit équivalent, nous avons appliqué la méthode automatique d'abstraction vue au chapitre 4. Ce circuit comporte trois variables d'état : le courant traversant l'inductance L_0, la tension aux bornes de C_1 (image de la force s'exerçant sur le ressort) et le courant traversant L_1 (image de la vitesse de déplacement de la masse). L'équation (58) correspond à l'équation d'état du circuit électrique équivalent au capteur de vibrations. L'équation (59) correspond à l'équation de sortie, seule la tension aux bornes de la résistance de charge a été définie comme étant une sortie du modèle. Le Tableau 16 présente les valeurs des composants qui ont été définies. Nous précisons que ces valeurs n'ont pas été choisies afin d'obtenir un modèle réaliste. Le but est ici de montrer que la méthode d'abstraction est toujours valide sur les systèmes électromécaniques. La Figure 45 montre une comparaison entre le modèle ELN, soit le circuit équivalent et le modèle LSF, soit la représentation d'état. Les deux courbes sont parfaitement superposées.

$$\begin{pmatrix} \dot{\imath}_{L1} \\ \dot{u}_{C1} \\ \dot{\imath}_{L0} \end{pmatrix} = \begin{pmatrix} -R_1/L_1 & -1/L_1 & \Psi/L_1 \\ k & 0 & 0 \\ -\Psi/L_1 & 0 & -(R_L+R_S)/L_0 \end{pmatrix} . \begin{pmatrix} i_{L1} \\ u_{C1} \\ i_{L0} \end{pmatrix} + \begin{pmatrix} R_1/L_1 \\ -k \\ \Psi/L_0 \end{pmatrix} . i_{IN} \tag{58}$$

$$u_e = (0 \quad 0 \quad R_L) . \begin{pmatrix} i_{L1} \\ u_{C1} \\ i_{L0} \end{pmatrix} \tag{59}$$

Tableau 16. Valeur des composants du circuit électrique équivalent au capteur de vibrations

Composant	Valeur
i_{IN}	1 A (Sinus)
L_1	10 H
R_1	5 KΩ
C_1	10 mF
ψ	10^{-6} N.A^{-1}
L_0	1 µH
R_S	100 mΩ
R_L	100 Ω

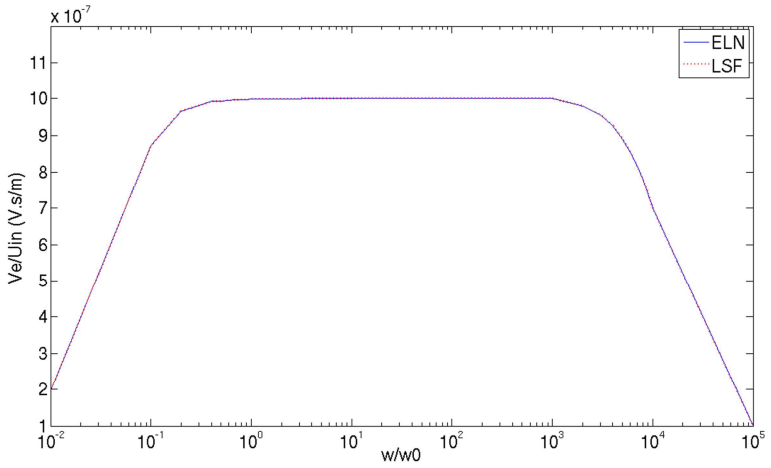


Figure 45. Comparaison entre le modèle LSF et le modèle ELN du capteur de vibrations

5.4.2 Filtre électromécanique

Les filtres électromécaniques sont utilisés dans les télécommunications pour leur facteur de qualité élevé. Ces systèmes présentent comme autres avantages d'avoir peu de pertes et d'obtenir un bon rapport signal à bruit. Ces filtres sont constitués d'un ou de plusieurs micro résonateurs. Le cas que nous nous proposons d'étudier par la suite est composé de deux micro-résonateurs associés. Ce système a donc deux degrés de liberté. Nous faisons l'hypothèse que les résonateurs sont identiques et symétriques. L'entrée de ce système est de type électrique, de même que la sortie, alors que le cœur du dispositif est lui de type mécanique. La Figure 46 présente le système que nous nous proposons d'étudier.

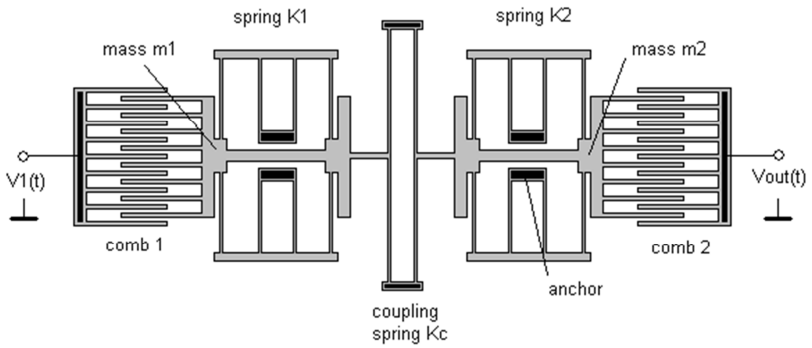


Figure 46. Filtre électromécanique constitué par une association de deux micro-résonateurs

Le signal électrique d'entrée est converti par un transducteur en vibrations mécaniques. Les deux dispositifs résonants filtrent ces vibrations puis un transducteur convertit les vibrations mécaniques en signal électrique. Les deux transducteurs sont de type électromécanique. Celui placé en entrée convertit une grandeur électrique en grandeur mécanique alors que celui placé en sortie convertit une grandeur mécanique en grandeur électrique. Le cœur du dispositif filtre les vibrations : une certaine bande de fréquence des oscillations mécaniques ne sera pas modifiée, ou très peu (en amplitude), alors que toutes les autres fréquences des oscillations mécaniques seront coupées. Le comportement d'un tel système est identique à celui d'un filtre passif passe bande du domaine électrique.

Etudions le dispositif. Tout d'abord le couplage électromécanique est effectué par une structure en peigne, en entrée comme en sortie du système. Les deux résonateurs sont couplés au moyen d'un ressort. Enfin, chaque résonateur est constitué d'une masse suspendue et d'un ressort auxquels un amortissement est ajouté. A partir de ces informations, nous sommes capables de déterminer le circuit équivalent électrique du filtre électromécanique qui est donné par la Figure 47. La partie de gauche, dont la référence est notée *gnd1* représente l'entrée électrique du système. Le couplage avec la partie mécanique est modélisé au moyen d'un transducteur électrostatique électrique/mécanique. La partie centrale, dont la référence est notée *gnd2* représente la partie mécanique du système. Le couplage avec la sortie électrique est modélisé au moyen d'un transducteur électrostatique mécanique/électrique. Enfin, la partie de droite, dont la référence est notée *gnd3* représente la sortie électrique du système.

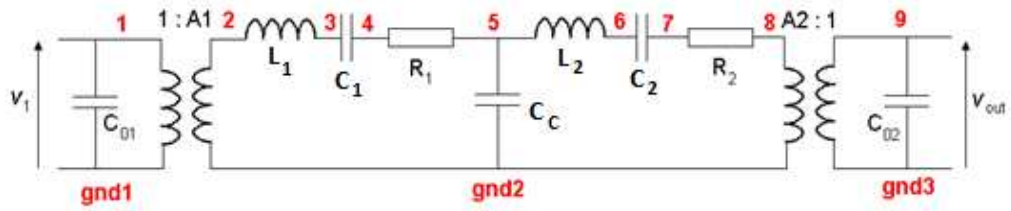


Figure 47. Filtre électromécanique constitué par une association de deux micro-résonateurs

Les deux résonateurs étant considéré symétriques et identiques, nous avons donc : $C_1 = C_2 = C_k$, $m_1 = m_2 = m$, $R_1 = R_2 = R$ et $C_{01} = C_{02} = C$. Nous supposons aussi les rapports de transduction égaux, donc $A_1 = A_2 = A$. Nous avons ensuite appliqué la méthode d'abstraction automatique afin de générer le modèle de haut niveau correspondant au filtre électromécanique. Ce circuit équivalent dispose de six variables d'état : le courant à travers l'inductance L_1 (vitesse de la masse m_1), le courant à travers l'inductance L_2 (vitesse de la masse m_2), la tension aux bornes de C_1 (force s'exerçant sur le ressort k_1), la tension aux bornes de C_2 (force s'exerçant sur le ressort k_2), la tension aux bornes de C_C (force s'exerçant sur le ressort k_C) et la tension aux bornes de C_{02}. La tension aux bornes de la capacité C_{01}

devrait théoriquement être considérée comme une variable d'état. Toutefois, étant donné que cette tension est la tension d'entrée du circuit, elle ne doit pas être ajoutée à la liste des variables d'état. L'équation (60) correspond à l'équation d'état du circuit équivalent au filtre électromécanique. L'équation (61) correspond à l'équation de sortie. Le seul signal du circuit qui a été défini comme une sortie est la tension V_{OUT}.

$$\begin{pmatrix} \dot{i_{L1}} \\ \dot{i_{L2}} \\ \dot{V_{C1}} \\ \dot{V_{C2}} \\ \dot{V_{CC}} \\ \dot{V_{OUT}} \end{pmatrix} = \begin{pmatrix} -R/L & 0 & -1/L & 0 & -1/L & 0 \\ 0 & -R/L & 0 & -1/L & 1/L & -A/L \\ C_k & 0 & 0 & 0 & 0 & 0 \\ 0 & C_k & 0 & 0 & 0 & 0 \\ C_C & -C_C & 0 & 0 & 0 & 0 \\ 0 & A/C & 0 & 0 & 0 & 0 \end{pmatrix} . \begin{pmatrix} i_{L1} \\ i_{L2} \\ V_{C1} \\ V_{C2} \\ V_{CC} \\ V_{OUT} \end{pmatrix} + \begin{pmatrix} A/L \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} . V_1 \tag{60}$$

$$V_{OUT} = (0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1) . \begin{pmatrix} i_{L1} \\ i_{L2} \\ V_{C1} \\ V_{C2} \\ V_{CC} \\ V_{OUT} \end{pmatrix} \tag{61}$$

Le Tableau 17 présente les valeurs des composants du circuit électrique équivalent au filtre électromécanique. Là encore, les valeurs n'ont pas été définies afin d'obtenir un modèle réaliste.

Tableau 17. Valeur des composants du circuit électrique équivalent au filtre électromécanique

Composant	Valeur
V_1	1 V (sinus)
C	1 nF
A	1
L	1 H
C_k	10 nF
R	1 KΩ
C_C	100 nF

Comme dans le cas d'étude précédent, nous avons effectué une comparaison des résultats de simulation entre le modèle ELN (circuit électrique équivalent) et le modèle LSF (représentation d'état). La Figure 48 montre que la courbe de réponse fréquentielle du modèle

mathématique est superposée à celle du modèle électrique. On remarque que l'on ne visualise qu'un seul pic de résonnance. Théoriquement un tel système a deux modes propres : le premier à la pulsation $\sqrt{1/C_k.L}$ et le second à la pulsation $\sqrt{1/(C_C + 2.C_k).L}$. Les deux modes apparaissent confondus car les valeurs des composants ont été choisies de telle sorte qu'ils sont très rapprochés. La Figure 49 présente un comparatif de la réponse indicielle du modèle ELN avec celle du modèle LSF. Ici aussi, les deux courbes sont superposées.

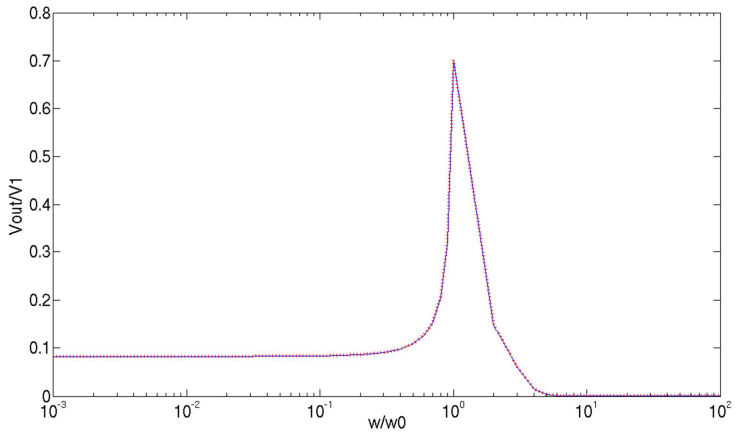


Figure 48. Comparaison entre le modèle LSF et le modèle ELN du filtre électromécanique

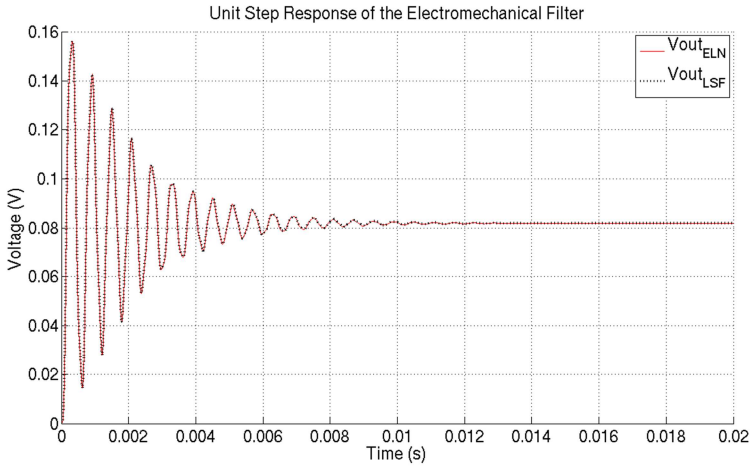


Figure 49. Comparaison réponse indicielle modèle ELN/modèle LSF pour un filtre électromécanique

Nous avons aussi effectué une analyse des durées de simulations. Le Tableau 18 présente les différents paramètres de simulation que nous avons choisis pour une analyse temporelle. Les calculs ont été effectués sur un PC Linux, CPU Intel Xeon X5570 (2.93 GHz, 8 Mo), RAM 24 Go (1066 MHz), HDD 10000 tours/min. Les durées de simulation sont les suivantes : 1092 ms pour le modèle ELN constitué du circuit équivalent électrique et 1010 ms pour le modèle LSF constitué de la représentation d'état, soit un écart de 8%.

Tableau 18. Paramètres de la simulation du filtre électromécanique

Fréquence de résonance	50,329 KHz
Fréquence du signal d'entrée	50,329 KHz
Durée simulée	397µs (20 périodes)
Pas de temps du simulateur	1.987 ns (10 000 points par période)
Résolution temporelle du simulateur	1.987 ns

Il est possible, comme dans le chapitre précédent de prendre en considération l'aspect puissance dans ce type de modélisation. Ce système absorbe et restitue des puissances électriques. Par contre, la partie active de ce système appartient au domaine mécanique : les pertes seront donc de type mécanique. Le bilan des puissances est en général très simple à établir pour les systèmes passifs. L'équation (62) montre le bilan des puissances du filtre électromécanique.

$$P_{OUT(ELEC)} = P_{IN(ELEC)} - P_{PERTES(MECA)} \tag{62}$$

Les seuls éléments capables de provoquer des pertes de type mécanique sont les éléments d'amortissement. Ici encore, nous retrouvons l'analogie avec l'électricité ou les pertes se concentrent au niveau des éléments résistifs sous forme de chaleur par l'effet Joule. Ainsi, pour modéliser les pertes de puissance dans ce type de systèmes, il faudra identifier le courant traversant les résistances (analogue à la vitesse de déplacement) et la tension à leurs bornes (analogue à la force qui s'applique). Afin de construire un modèle du même type que celui construit au chapitre précédent pour les circuits analogiques passifs, nous allons donc insérer dans le vecteur de sortie de la représentation d'état le courant et la tension d'entrée du filtre (puissance électrique fournie) ainsi que le couple courant-tension correspondant à chaque résistance (puissance des pertes mécaniques).

L'équation (63) présente le vecteur de sortie de la représentation d'état modifié en y incluant les informations nécessaires au calcul des puissances mises en jeu.

$$\begin{pmatrix} V_{OUT} \\ V_{R1} \\ V_{R2} \\ I_{IN} \\ I_{R1} \\ I_{R2} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ R_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & R_2 & 0 & 0 & 0 & 0 \\ 1/A & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} . \begin{pmatrix} i_{L1} \\ i_{L2} \\ V_{C1} \\ V_{C2} \\ V_{CC} \\ V_{OUT} \end{pmatrix} \quad (63)$$

A partir de cette équation, nous sommes capables de produire une analyse de puissance sur le filtre électromécanique. La Figure 50 présente un bilan des puissances du filtre électromécanique. La courbe apparaissant en bleu est la puissance électrique absorbée, elle est donnée par le produit de la tension d'entrée et du courant d'entrée comme spécifié dans l'équation (64).

$$P_{IN(ELEC)} = V_{IN} . I_{IN} \quad (64)$$

Les pertes mécaniques apparaissent en rouge. Elles sont données par l'équation (65). Les pertes sont égales à la somme des puissances dissipées par les éléments d'amortissement.

$$P_{PERTES(MECA)} = V_{R1} . I_{R1} + V_{R2} . I_{R2} \quad (65)$$

Sur la Figure 50, il apparaît clairement que presque aucune puissance n'est consommée lorsque la fréquence du signal d'entrée se trouve en dehors de la bande passante du filtre. A l'intérieur de la bande passante, la partie mécanique entre en résonnance. Une grande partie de la puissance est dissipée dans les éléments d'amortissement. On retrouve ici le comportement typique d'un filtre électrique RLC. A la résonnance il est équivalent à la seule résistance, ceci entrainant une surintensité et donc une puissance importante dissipée sous forme d'effet Joule.

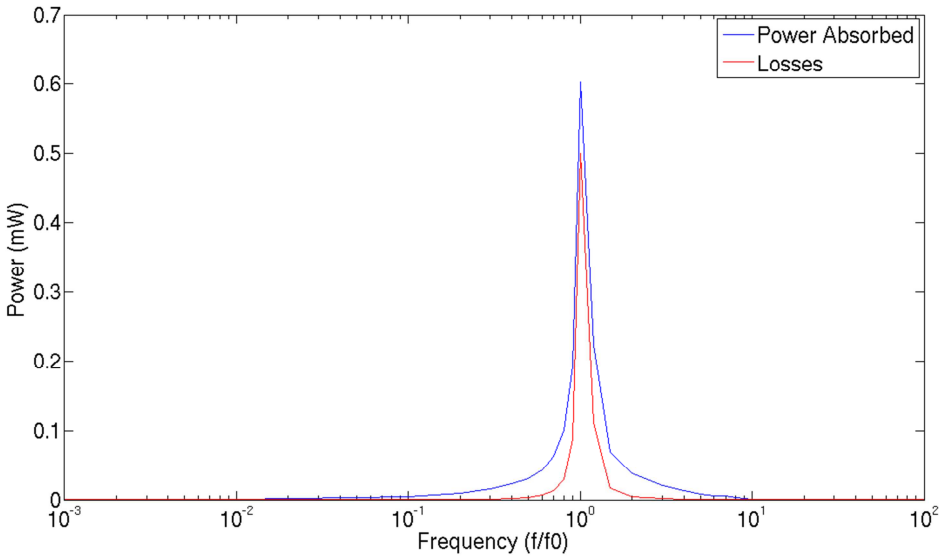


Figure 50.Bilan des puissances du filtre électromécanique

5.5 Les systèmes à paramètres distribués

Un système est dit à paramètres distribués lorsqu'on ne peut pas faire l'hypothèse que la longueur d'onde du signal est très petite devant les dimensions physiques du système. Il faut alors appliquer la théorie de la propagation des ondes. Toutefois, les comportements dynamiques étant similaires, on considère souvent qu'un modèle à paramètres distribués est un raffinement du modèle à paramètres localisés. Prenons l'exemple d'une corde de guitare : après avoir été pincée, elle revient à sa position d'équilibre. Si l'on se place dans l'hypothèse des paramètres localisés, on considère que la corde revient à sa position d'équilibre en vibrant à la fréquence fondamentale uniquement, celle de la note jouée (exemple : La440). Si l'on se place dans l'hypothèse des paramètres distribués, alors on considère que la corde revient à sa position d'équilibre en vibrant à une combinaison de différentes fréquences (La440 + La880 + La1320, etc...). Ces différentes fréquences sont les harmoniques, ils sont appelés modes de vibration. Le premier mode correspond à la fréquence fondamentale. Ainsi la vibration correspondant à chacun des modes contribue à la vibration totale. Le diagramme de la Figure 51 illustre la problématique des systèmes à paramètres distribués. Alors que dans un système à paramètres localisés tel que le capteur de vibrations, il y a deux domaines physiques à prendre en compte, dans un système à paramètres distribués, introduire un mode de vibrations revient – en pratique – à introduire un domaine physique. Si l'on considère un système bâti autour d'un transducteur électrique vers mécanique, il faudra ainsi ajouter un élément de transformation pour passer du domaine électrique à un mode de vibrations mécaniques, déterminer ensuite le circuit équivalent électrique de chacun des modes puis, enfin calculer la contribution de chaque mode au domaine mécanique global en disposant là aussi un élément de transformation.

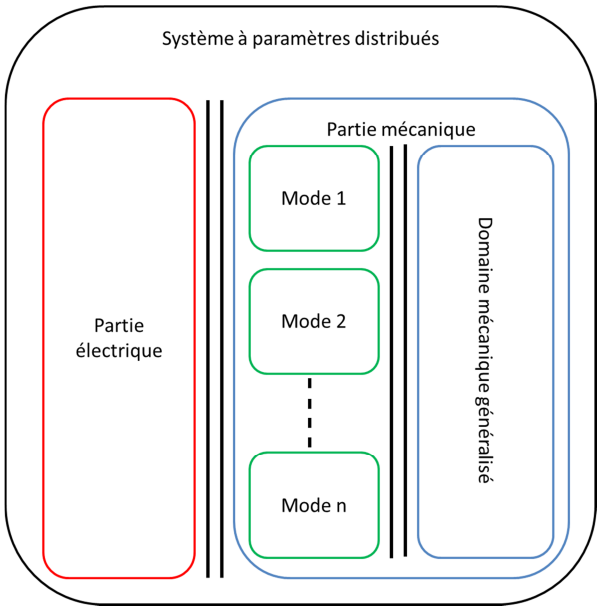


Figure 51.Diagramme d'un système à paramètres distribués

5.6 Modélisation d'un capteur de force électrostatique

La Figure 52 présente un capteur de force, cet exemple est tiré de [Tilmans 97]. Il utilise un transducteur électrostatique. Nous nous proposons de générer le modèle électrique correspondant à ce capteur de force. Il est constitué d'une poutre de section rectangulaire travaillant en flexion. La longueur *l* de la poutre est plus grande que toutes les autres dimensions. Les ondes se propagent suivant les 3 axes mais la propagation suivant l'axe *x* prévaut largement. Ainsi, nous allons négliger les déplacements suivant les autres axes et nous ramener à un problème à une dimension. Le couplage électrostatique a lieu aux frontières du système, au niveau des électrodes. Nous considérons que les surfaces des électrodes sont uniformément recouvertes par un matériau conducteur, dont la masse et la raideur sont négligeables. Cette poutre est en position horizontale, elle est encastrée à ses deux extrémités. Une force verticale que l'on souhaite mesurer est appliquée en un point de cette poutre.

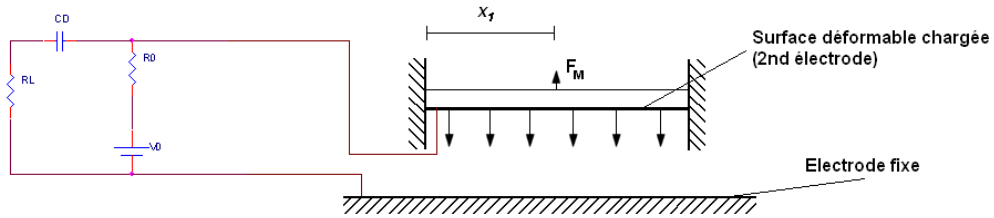


Figure 52. Représentation du capteur de force

5.6.1 Description du modèle du capteur de force

L'objectif n'est pas pour nous de déterminer un modèle analytique. Dans l'industrie, des ingénieurs spécialisés dans les MEMS (physiciens) établissent de tels modèles. Notre objectif est ici de générer automatiquement le circuit équivalent électrique au capteur de force à partir des données et des équations fournies par les spécialistes. Le Tableau 19 présente les expressions des éléments du circuit équivalent électrique. *l*, *b* et *h* correspondent respectivement à la longueur de la poutre, la largeur de la poutre et l'épaisseur de la poutre. La Figure 53 présente la forme du circuit équivalent au capteur de force.

C_0 représente la capacité du transducteur électrostatique, R_n (c : coefficient de trainée) l'amortissement correspondant à un mode, K_n la compliance associée à un mode, K'_n la compliance correspondant au couplage électromécanique pour un mode, K^*_n la résultante des deux compliances précédentes, M_n la masse associée à un mode, I le moment d'inertie second de la poutre, λ_n la constante correspondant au mode *n*, ω_n^* la pulsation d'un mode en présence d'un couplage électromécanique, Φ_n la fonction propre d'un mode, χ_{1n} le facteur a appliquer à un mode et Γ_n le facteur de transduction correspondant à un mode.

Sur le circuit de la Figure 53 ainsi dans le Tableau 19, certaines valeurs sont annotées avec *. Cette annotation apparaît en cas de couplage électromécanique. Par exemple, $\Phi_n(x)$ représente la fonction propre d'une poutre encastrée à ses deux extrémités en l'absence de couplage électromécanique [Meirovitch 75]. $\Phi_n^*(x)$ représente la fonction propre d'une poutre encastrée à ses deux extrémités en présence d'un couplage électromécanique. Dans le cas du capteur de force, le couplage électromécanique de type électrostatique est évident. Cependant, $\Phi_n(x)$ constitue une très bonne approximation de $\Phi_n^*(x)$. La condition nécessaire pour pouvoir faire cette approximation est que l'espace entre la poutre et l'électrode fixe soit très grand devant l'amplitude des vibrations de la poutre [Tilmans 97].

Toutefois, si les fonctions propres en présence de couplage peuvent être approximées par les expressions sans couplage, il n'en est pas de même pour les pulsations propres ω_n^*. En effet, le couplage électromécanique induit l'ajout d'un ressort de constante de raideur K'_n. La constante de raideur totale est donc diminuée de K'_n.

Tableau 19. Expressions des éléments du circuit équivalent électrique

Elément	Expression
C_0	$b\int_0^l \frac{\varepsilon_0}{d+w_0(x)}dx \approx \frac{\varepsilon_0.b.l}{d}$
R_n	$\int_0^l c.{\Phi_n}^2(x)dx = cl$
K_n	$\int_0^l \Phi_n(x)\mathcal{L}[\Phi_n(x)]dx = {\omega_n}^2.M_n$
${K'}_n$	$\int_0^l \frac{b.\varepsilon_0.{V_0}^2[\Phi_n(x)]^2}{(d+w_0(x))^3}dx \approx \frac{b.l.\varepsilon_0.{V_0}^2}{d^3}$
${K^*}_n$	$K_n - {K'}_n$
M_n	$\int_0^l \rho.b.h.{\Phi_n}^2(x)dx = \rho.b.h.l = m$
I	$\frac{b.h^3}{12}$
$\lambda_n.l$	$(2.n+1).\frac{\pi}{2}$
$(\omega_n)^2$	$\frac{E.I.{\lambda_n}^4}{\rho.b.h}$
$({\omega_n}^*)^2$	$({\omega_n}^*)^2 = \frac{{K^*}_n}{M_n}$
$\Phi_n(x)$	$\cos(\lambda_n.x) - \cosh(\lambda_n.x) + \frac{\cos(\lambda_n.l) - \cosh(\lambda_n.l)}{\sinh(\lambda_n.l) - \sin(\lambda_n.l)}(\sin(\lambda_n.x) - \sinh(\lambda_n.x))$
χ_{1n}	$\frac{1}{l}\int_0^l \Phi_n(x)dx$
Γ_n	$\int_0^l \frac{b.\varepsilon_0.V_0.{\Phi^*}_n(x)}{(d+w_0(x))^2}dx \approx \frac{\varepsilon_0.b.l.V_0}{d^2}.\chi_{1n}$

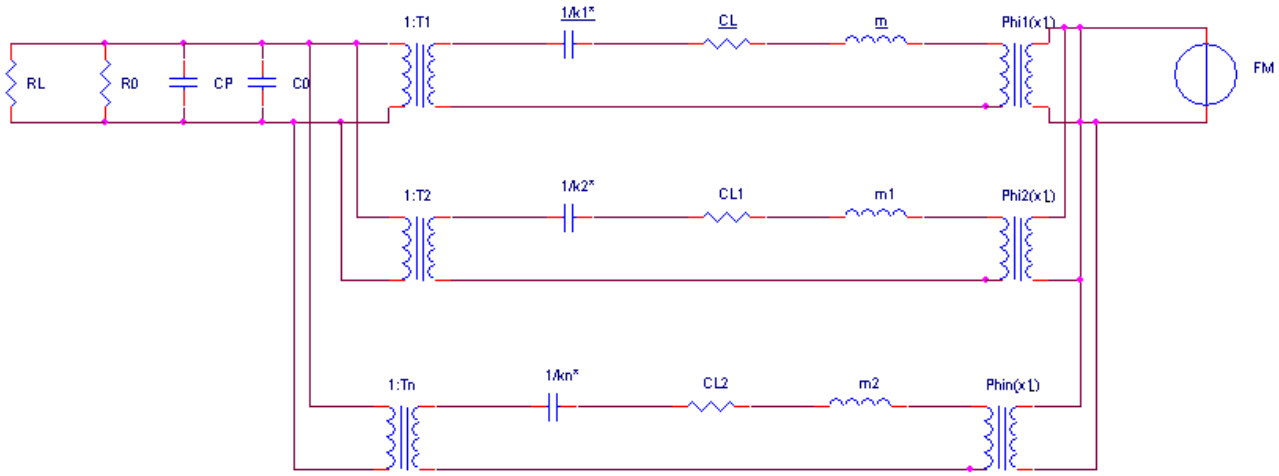


Figure 53. Circuit équivalent au capteur de force

Nous remarquons que la topologie du circuit équivalent est identique pour chacun des modes. Les valeurs de l'inductance et de la résistance modélisant respectivement la masse et l'amortissement sont aussi identiques pour chacun des modes. Nous avons donc crée un module ELN correspondant à ce circuit : à partir du transformateur dont le rapport est Γ_n jusqu'au transformateur dont le rapport est Φ_n. Ce module est un quadripôle. Il reçoit les valeurs de chacun des éléments par paramètres. Ces paramètres sont de type `double.` Le code en Annexe 2.2 présente le module permettant de modéliser un mode : `mod_eec`.

Afin de créer le modèle global du capteur, le module précédent est instancié *n* fois (mais avec des paramètres `sigma_v`, `phi_v` et `comp_v` différents) entre le domaine électrique et le domaine mécanique généralisé , *n* correspondant au nombre de modes que l'on souhaite prendre en compte. La liste des paramètres que l'utilisateur doit spécifier pour utiliser la classe `force_sensor` est donnée par le Tableau 20. La valeur ε_0 est définie en constante et ne doit pas être spécifiée par l'utilisateur.

Tableau 20. Paramètres de la classe `force_sensor`

Nom	Type	Description
nm	sc_core::sc_module_name	Nom du module
force	sca_tdf::sca_signal <double>	Signal TDF de la force à mesurer
position	double	Position où s'applique la force à mesurer
modes	int	Nombre de modes à prendre en compte
length	double	Longueur de la poutre
width	double	Largeur de la poutre
thickness	double	Epaisseur de la poutre
gap	double	Espace entre les électrodes au repos
drag	double	Coefficient de trainée
bias	double	Tension de polarisation
rho	double	Masse volumique du matériau qui constitue la poutre
young_mod	double	Module d'Young du matériau qui constitue la poutre

5.6.2 Simulation du modèle de capteur de force

Le modèle a été simulé sur la même machine que celle utilisée précédemment : PC Linux, CPU Intel Xeon X5570 (2.93 Ghz, 8 Mo), RAM 24 Go (1066 MHz), HDD 10000 tours/min, SystemC AMS 1.0 bêta 1, SystemC 2.2. Le Tableau 21 présente les paramètres de la simulation.

Tableau 21. Paramètres de la simulation du capteur de force

Durée simulée	200 ms
Pas de temps du simulateur	1 µs
Résolution temporelle du simulateur	1 µs

Les trois premiers modes ont été simulés, ce qui implique de résoudre un nombre total de 28 équations avant de réellement créer le circuit. Ces 28 équations correspondent aux calculs des expressions des éléments du Tableau 19, elles sont résolues par un seul solveur TDF. La durée nécessaire pour résoudre ces équations est faible et surtout, elle restera constante quelle que soit la durée simulée. Il est donc évident que plus la durée simulée sera longue, plus le calcul des valeurs des éléments du circuit représentera une partie négligeable de la durée totale de la simulation. La durée de simulation totale (élaboration + simulation) est dans le cas présent de 1218 ms (valeur moyenne obtenue sur dix essais). La Figure 54 présente un résultat de simulation du capteur de force. La Figure 55 présente un agrandi sur la zone [145 ms ; 181 ms]. Nous observons la réponse du capteur à une force appliquée pendant 2 ms. Une forme de non linéarité apparaît puisque la tension aux bornes de la résistance de sortie est une somme de sinus. Chaque mode de vibration contribue de manière plus ou moins importante au signal global. Après l'impulsion, nous voyons que c'est le mode fondamental qui domine. Mais juste avant que la poutre ne rejoigne sa position d'équilibre, à partir de 175 ms, nous voyons que le mode fondamental ne contribue plus au signal global mais que le troisième mode subsiste.

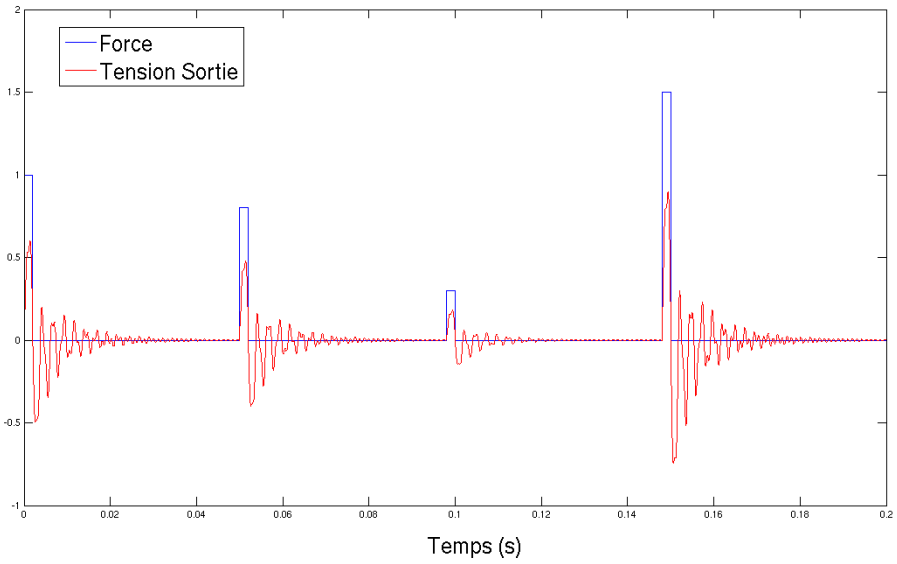


Figure 54. Simulation du capteur de force

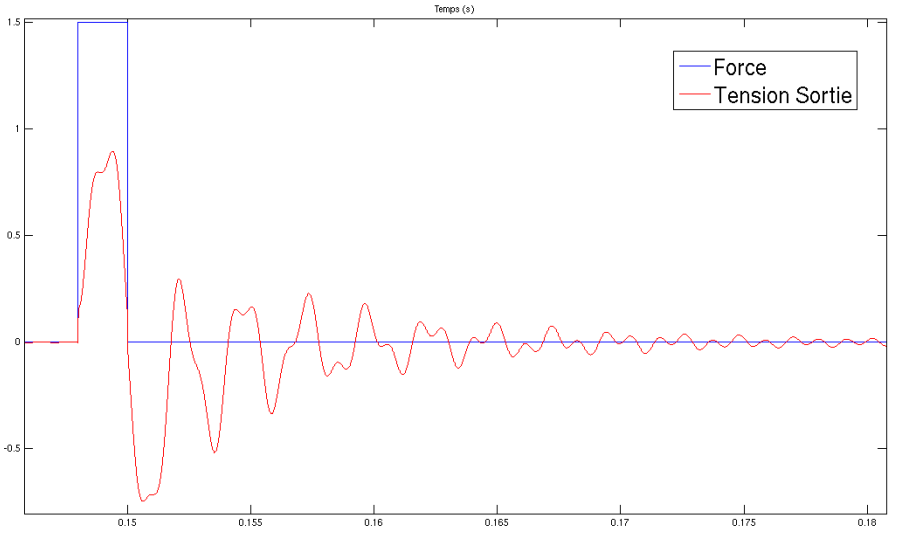


Figure 55. Simulation du capteur de force (zoom)

5.6.3 Seconde approche : élever le niveau d'abstraction de départ

Le paragraphe précédent présentait une classe C++ permettant de générer automatiquement le circuit électrique équivalent d'un capteur de force à partir de ses propriétés physiques : dimensions de la poutre, masse volumique et module d'Young du matériau, etc… Une seconde approche consiste à fournir directement les valeurs des éléments du circuit

équivalent. Ces valeurs peuvent être passées en paramètres facilement au moment de l'instanciation au moyen du type `sca_util::sca_vector <double>`. L'instanciation devient beaucoup plus légère à réaliser pour le modélisateur :

```
force_sensor fs("fs",order_v,phi_v,sigma_v,comp_v,mass_v,damp_v) ;
//declaration,instanciation
fs.p_meca(mecap);//connexions
fs.n_meca(mecaref);
fs.p_elec(elecp);
fs.n_elec(elecref);
```

De même, le modèle du capteur de force ne comporte plus aucun calcul. Seules les instanciations des circuits équivalents à chaque mode de vibration (`mod_eec`) sont réalisées. Le code en annexe 2.3 présente le modèle du capteur de force tel qu'il devient si les paramètres utilisés ne sont plus les propriétés physiques de la poutre mais directement les valeurs des éléments du circuit équivalent.

Ce module est très simple. Une question se pose alors : quel modèle est préférable ? Le physicien qui conçoit le capteur a une approche très bas niveau, le modèle proposé au paragraphe précédent a plus de chances de lui convenir. Toutefois, il n'est probablement pas un utilisateur naturel de SystemC AMS, il travaille plutôt sur des logiciels de simulation par éléments finis du type ANSYS. Si l'on se place maintenant du point de vue des ingénieurs qui souhaitent simuler le comportement global d'un système ou développer son logiciel embarqué, on s'aperçoit que ceux-ci désirent simplement disposer d'un prototype virtuel, le plus simple et le plus léger étant souvent considéré comme le meilleur (si les deux modèles sont équivalents en termes de comportement). C'est donc le modèle où ce sont les valeurs des éléments du circuit équivalent qui sont passées en paramètres qui lui conviendra le mieux.

5.6.4 Génération du modèle de haut niveau

Ici, nous avons exécuté notre méthode permettant de générer automatiquement la représentation d'état du circuit électrique équivalent du capteur de force. Si l'on ne prend en compte que le premier mode de vibrations, l'équation d'état est donnée par l'équation et l'équation (66) de sortie est donnée par l'équation (67).

$$\begin{pmatrix} \dot{\imath_{m1}} \\ \dot{V_{K1}} \\ \dot{V_{C0}} \end{pmatrix} = \begin{pmatrix} -R/m & -1/m & -\Gamma_1/m \\ 0 & 0 & K_1 \\ 1/\Gamma_1.C_0 & 0 & -1/R \end{pmatrix} . \begin{pmatrix} i_{m1} \\ V_{K1} \\ V_{C0} \end{pmatrix} + \begin{pmatrix} \Phi_1/m \\ 0 \\ 0 \end{pmatrix} . V_{Force} \tag{66}$$

$$V_{RL} = (0 \quad 0 \quad 1) . \begin{pmatrix} i_{m1} \\ V_{K1} \\ V_{C0} \end{pmatrix} \tag{67}$$

Si l'on ajoute le second mode de vibrations, alors la représentation d'état est donnée par les équations (68) et (69).

$$\begin{pmatrix} \dot{\imath_{m1}} \\ \dot{V_{K1}} \\ \dot{V_{C0}} \\ \dot{\imath_{m2}} \\ \dot{V_{K2}} \end{pmatrix} = \begin{pmatrix} -R/m & -1/m & -\Gamma_1/m & 0 & 0 \\ K_1 & 0 & 0 & 0 & 0 \\ 1/\Gamma_1.C_0 & 0 & -1/R & 1/\Gamma_2.C_0 & 0 \\ 0 & 0 & -\Gamma_2/m & -R/m & -1/m \\ 0 & 0 & 0 & K_2 & 0 \end{pmatrix} . \begin{pmatrix} i_{m1} \\ V_{K1} \\ V_{C0} \\ i_{m2} \\ V_{K2} \end{pmatrix} + \begin{pmatrix} \Phi_1/m \\ 0 \\ 0 \\ \Phi_2/m \\ 0 \end{pmatrix} . V_F \tag{68}$$

$$V_{RL} = (0 \quad 0 \quad 1 \quad 0 \quad 0) . \begin{pmatrix} i_{m1} \\ V_{K1} \\ V_{C0} \\ i_{m2} \\ V_{K2} \end{pmatrix} \tag{69}$$

Lorsque nous complexifions le modèle en ajoutant un mode de vibrations, nous ajoutons une instance de `mod_eec`, le circuit électrique équivalent d'un mode. L'ajout de cette instance est fait en parallèle des (ou du) modules `mod_eec` déjà présents. Ainsi, le fait d'ajouter une instance n'introduit pas de problème de couplage : les modes sont totalement indépendants les uns des autres. Nous savons donc comment est modifiée la représentation d'état si l'on ajoute un mode. Il nous est donc possible de générer la représentation d'état sans même utiliser le circuit électrique équivalent comme intermédiaire. Les équations successives (70), (71), (72) présentent les équations d'état que l'on peut générer automatiquement. L'équation (73) présente l'équation de sortie.

$$\dot{\imath_{mi}} = \frac{-R}{m} . i_{mi} - \frac{1}{m} . V_{Ki} - \frac{V_{C0}}{m} . \Gamma_i + \frac{V_F}{m} . \Phi_i \tag{70}$$

$$\dot{V_{Ki}} = K_i . i_{mi} \tag{71}$$

$$\dot{V_{C0}} = \frac{-1}{R} . V_{C0} + \frac{1}{C_0} \sum_{1}^{n} \frac{i_{mi}}{\Gamma_i} . \tag{72}$$

$$V_{RL} = V_{C0} \tag{73}$$

Ces équations constituent la représentation d'état du capteur de force. Il n'est pas nécessaire de générer la netlist correspondant au circuit équivalent électrique et d'appliquer la méthode d'abstraction afin de la déterminer.

5.7 Conclusion du chapitre

Dans ce chapitre, nous avons vu qu'il est possible de modéliser des systèmes électromécaniques à haut niveau d'abstraction avec SystemC AMS. Cela signifie qu'une fois que le modèle très détaillé des capteurs ou des actionneurs a été développé par les physiciens, il est possible de réaliser des simulations globales en incorporant les modèles de haut niveau

correspondant aux parties physiques dans le modèle global. Les parties numériques matérielles et logicielles, les parties analogiques et les parties capteurs et actionneurs pourront donc être simulées ensemble en des temps relativement faibles au moyen de SystemC AMS.

Nous avons utilisé ici une approche bottom-up où le circuit électrique équivalent est un modèle intermédiaire. Les spécialistes de la physique travaillent à très bas niveau sur des outils d'analyse par éléments finis. Ils produisent des modèles analytiques qui peuvent être traduits en circuits électriques équivalents. A partir d'un circuit équivalent, nous sommes capables de produire un modèle mathématique de haut niveau simulable en un temps réduit. Dans le cas des systèmes à paramètres distribués, ajouter un mode de vibrations se traduit par une instance en parallèle d'un bloc supplémentaire. La topologie de ce bloc demeure la même, quel que soit le mode de vibrations, seules les valeurs de certains de ces éléments sont modifiées. Il a été possible, dans ce cas : de générer la représentation d'état directement, à partir du modèle analytique (équations) sans passer par le circuit équivalent et la méthode d'abstraction présentée au chapitre 4.

Nous avons aussi vu que tout comme dans le cas des circuits électriques, il est possible d'inclure dans les modèles de haut niveau des systèmes électromécaniques des informations permettant de calculer puis de simuler les puissances mises en jeu.

6. Les Bond Graphs

Au cours du chapitre précédent, nous avons vu que l'on pouvait modéliser les parties mécaniques d'un système par un circuit électrique. Ceci est aussi possible pour d'autres domaines de la physique comme l'acoustique et l'hydraulique, par exemple. Introduite en 1959 par Henry M. Paynter, la méthode de modélisation par les bond graphs est une approche différente. Elle constitue plutôt une approche unifiée et neutre. Plutôt que de ramener les éléments de tous les domaines physiques à l'électrique, tous les domaines physiques, y compris l'électrique sont modélisés par les bond graphs. Cette méthode a été raffinée par Dean C. Karnopp et Ronald C. Rosenberg, étudiants de Paynter. Ce formalisme a été appliqué à différents domaines de la physique et il s'est avéré être efficace pour les systèmes multiphysiques. De nos jours, cette approche unifiée est de plus en plus utilisée dans tous les domaines de la physique. Un grand nombre d'outils ont été développés afin de pouvoir créer des modèles bond graph et les simuler, par exemple : 20-sim, ARCHER, MS1 ou encore une toolbox pour Mathematica (une liste exhaustive peut être trouvée sur [Bondgraphs 13]).

Torsten Mähne a produit un MoC SystemC AMS dédié aux bond graphs [Mähne 11] et la bibliothèque SCAX correspondante (cette bibliothèque n'a pas été rendue publique à ce jour). Il serait donc intéressant de disposer de méthodes permettant de générer automatiquement des modèles bond graphs et de pouvoir les simuler sous SystemC AMS. Dans [Lichiardopol 07], une méthode permettant de générer des bond graphs à partir de données expérimentale est proposée. C'est donc naturellement que nous avons cherché à développer une méthode permettant de générer un bond graph à partir de modèles de bas niveau. Ce chapitre débute par une introduction à la méthodologie bond graph en présentant ses différents concepts. Nous présentons ensuite la méthode de construction systématique d'un bond graph ainsi que les règles d'affectations de la causalité et enfin comment il est possible de déduire le modèle mathématique correspondant à un bond graph. Enfin, nous présentons la classe Netlist2bg, qui permet de générer automatiquement un modèle bond graph à partir d'un modèle de bas niveau.

6.1 Généralités sur les bond graphs

Les bond graphs (graphes de liens) sont basés sur une approche énergétique des systèmes. Les principes de l'analogie électromécanique présentés au début du chapitre précédent peuvent se résumer par le tétraèdre de Paynter qui apparaît sur la Figure 56. On y voit sur les quatre sommets du carré les variables de puissance e et f : l'effort et le flux et les variables d'énergie p et q : le moment et le déplacement. Il apparaît sur ce schéma que le moment et le déplacement sont obtenus respectivement par intégration par rapport au temps de l'effort et du flux comme spécifié dans l'équation (55) et dans l'équation (56). Apparaissent aussi sur cette représentation trois variables : R, C et I qui lient respectivement l'effort et le flux, l'effort et le déplacement, le flux et le moment.

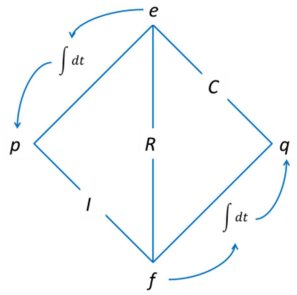


Figure 56. Tétraèdre de Paynter

L'élément *R* permettra de modéliser sur un bond graph une résistance électrique ou une résistance visqueuse, l'élément *C* une capacité ou un ressort, l'élément *I* une inductance ou une masse (Tableau 11, chapitre précédent). L'élément *R* est appelé élément dissipatif. Les éléments *C* et *I* sont des éléments de stockage. Plus précisément, dans la littérature, *C* peut être appelé élément de stockage d'énergie potentielle et *I* élément de stockage d'énergie cinétique. Les éléments *R*, *C* et *I* constituent les briques de base des bond graphs, de la même façon que la résistance, la capacité et l'inductance sont les briques de base de l'électricité. Il existe aussi des éléments actifs ou sources qui fournissent de la puissance au système. La source d'effort, notée *Se* permet de modéliser, entre autres, une source de tension. La source de flux, notée *Sf* permettra quant à elle de modéliser notamment une source de courant. Enfin, il existe aussi des éléments de jonction. Ils sont notés *0*, *1*, *TF* (transformateur) et *GY* (gyrateur). Ils sont utilisés pour coupler les éléments. Ils sont conservatifs de puissance. Le Tableau 22 présente un récapitulatif des différents éléments pouvant constituer un bond graph.

Tableau 22. Les différents éléments des bond graphs et ce qu'ils modélisent

Elément	Exemples
R	Résistance électrique, Résistance visqueuse
C	Capacité, Ressort
I	Inductance, Masse
Se	Source de tension, Source de pression
Sf	Source de courant
0	Couplage série en mécanique, Couplage parallèle en électricité
1	Couplage parallèle en mécanique, Couplage série en électricité
TF	Transformateur électrique, Levier, poulies, engrenages, Couplage entre domaines physiques
GY	Moteur à courant continu, Couplage entre domaines physiques

La méthode bond graph étant basée sur les échanges d'énergie, le bond graph est un graphe orienté. Un lien entre deux systèmes est représenté par une demi-flèche, comme cela apparaît sur la Figure 57. La demi flèche indique le sens du transfert de la puissance.

$$S_1 \xrightarrow[f(t)]{e(t)} S_2$$

Figure 57. Formalisme Bond graph : lien entre deux systèmes

6.2 Construction d'un modèle Bond graph

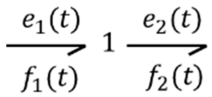
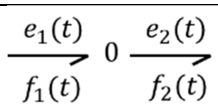
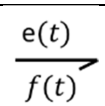
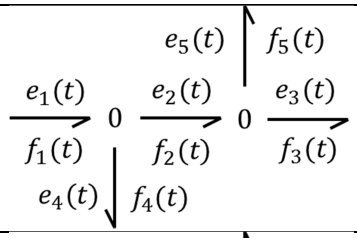
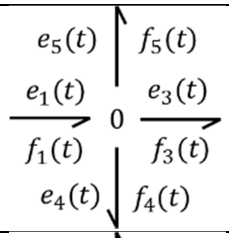
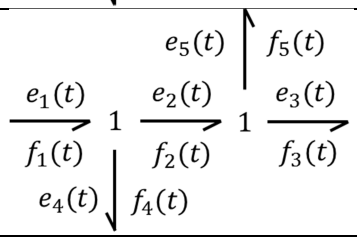
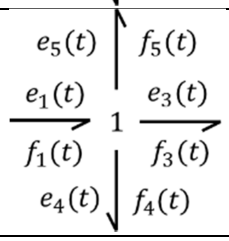
Une méthode systématique permettant de construire un Bond graph existe. Elle a été présentée dans la littérature [Dauphin 99].Il y a un principe fondamental : deux jonctions du même type ne peuvent pas être liées directement. Comme nous l'avons vu au cours du chapitre précédent, il y a une dualité entre le domaine mécanique et le domaine électrique, ainsi, la méthode diffère selon le domaine. La première étape consistera donc à identifier clairement les domaines physiques et les éléments s'y rapportant.

Ensuite, pour chaque nœud électrique une jonction 0 est créée. Dans le domaine électrique, une jonction 0 correspond à un potentiel électrique. Pour chaque différence de potentiel introduite par un composant il faut introduire une jonction 1 entre les deux jonctions 0 correspondantes aux potentiels et y attacher ledit composant, par l'intermédiaire d'une nouvelle liaison 0 correspondant à la différence de potentiel. Les jonctions peuvent ensuite être reliées par des liens, en respectant le sens du transfert d'énergie.

Dans le domaine mécanique, pour chaque vitesse une jonction 1 est créée. Un élément est attaché à la jonction 1 correspondante à sa vitesse. Ensuite, pour chaque différence de vitesse une jonction 0 est créée afin de pouvoir lier les deux jonctions 1 correspondantes.

Enfin, il est possible de simplifier le bond graph en suivant les règles de simplification présentées dans le Tableau 23. L'algorigramme de la Figure 58 présente la méthode systématique de construction d'un modèle bond graph. L'étape de fusion du bond graph de la partie électrique et du bond graph de la partie mécanique (encadrée en rouge) est particulière : les éléments de couplage se situant à la frontière entre les deux domaines, ils ont été traités à la fois dans le domaine électrique et dans le domaine mécanique. Ils étaient alors considérés comme des éléments à un port, ne pouvant être reliés qu'à une seule jonction. Il s'agira donc au cours de l'étape de fusion de les considérer comme des éléments à deux ports, ce qui permettra de connecter la partie mécanique avec la partie électrique.

Tableau 23. Règles de simplification

Forme non simplifiée	Forme simplifiée
	
	
	
	

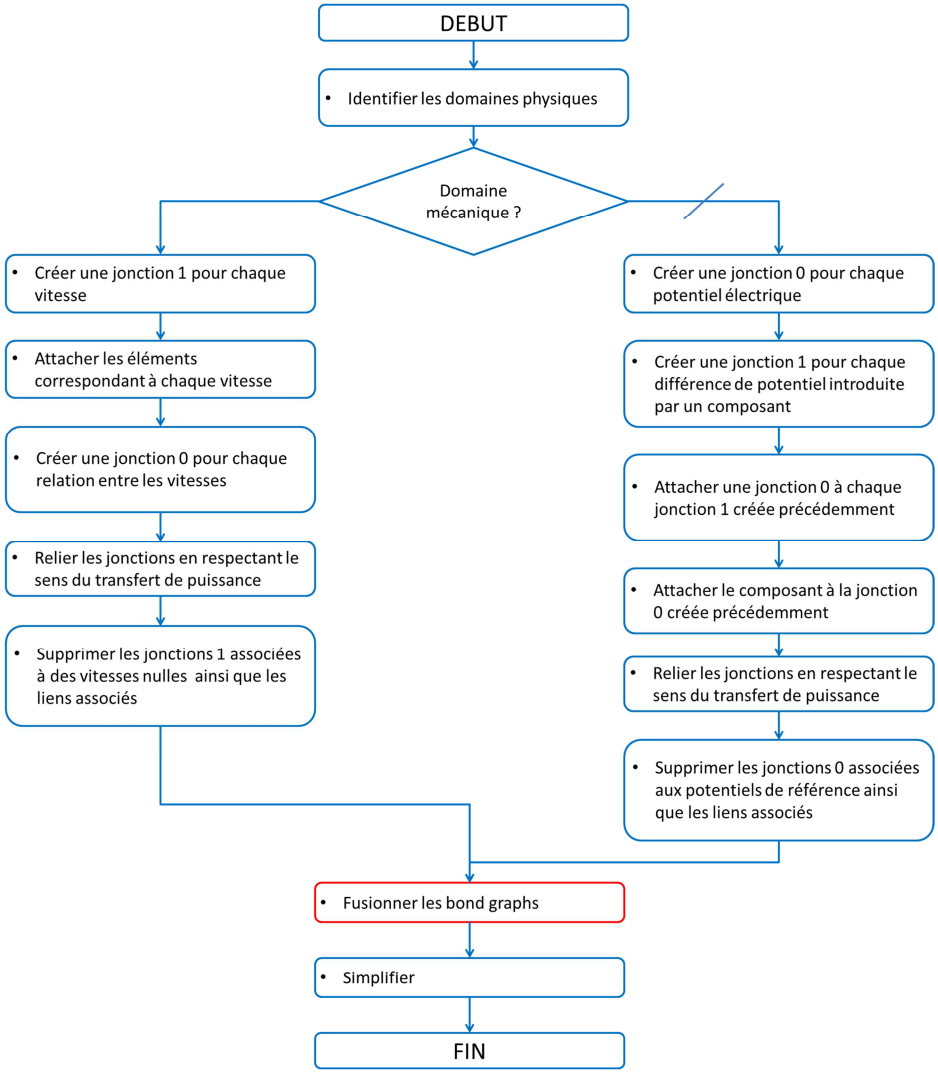


Figure 58. Méthode de construction du bond graph d'un système électromécanique

6.3 Exemple : le moteur à courant continu associé au réducteur

Voyons un exemple faisant intervenir les domaines mécaniques et électriques : un moteur électrique associé à un réducteur. Ce système est présenté par la Figure 59. Le moteur électrique est alimenté par une source de tension. Le modèle équivalent d'un moteur à courant continu est constitué par l'association série d'une résistance et d'une inductance. Le couplage entre le domaine électrique et le domaine mécanique est donné par la loi $\omega_1 = K.e$. C'est une relation entre un effort (e) et un flux (ω_1). La résistance est modélisée en bond graph par l'élément *R* et l'inductance par l'élément *I*. C'est le gyrateur *GY*, de rapport *K* qui va nous permettre de modéliser le couplage électromécanique. Le réducteur est quant à lui constitué d'un engrenage formé de deux roues dentées. Ces éléments ont une masse non négligeable : ils introduiront donc tous deux un élément de type inertiel *I*. La réduction se fait par un élément de transformation suivant la loi $\omega_1 = r.\omega_2$, nous utiliserons donc un élément *TF*, de rapport *r*.

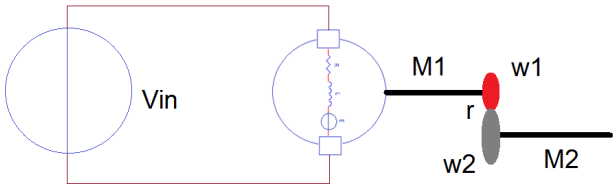


Figure 59. Moteur électrique à courant continu couplé à un réducteur

La Figure 60 montre le bond graph non simplifié obtenu suite aux opérations décrites au cours du paragraphe précédent. Dans le domaine électrique, nous voyons bien que chaque liaison 0 correspond à un potentiel ou à une différence de potentiel. Les éléments sont connectés à la liaison 0 correspondant à la différence de potentiel qu'ils introduisent. Entre chaque liaison 0, nous avons une liaison 1. Dans le domaine mécanique, chaque vitesse est associée à une liaison 1, à laquelle est attaché l'élément correspondant.

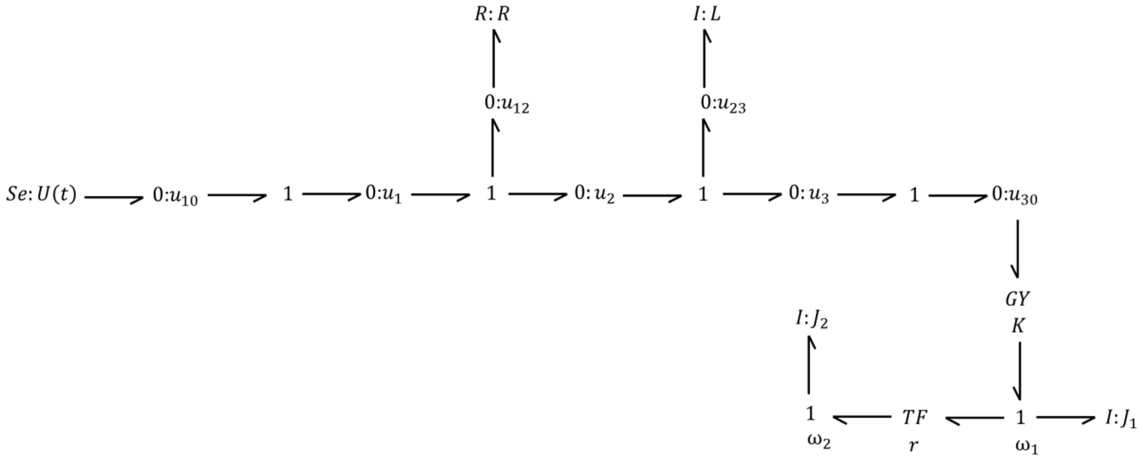


Figure 60. Bond graph non simplifié du moteur associé au réducteur

La Figure 61 présente le bond graph simplifié du système. Nous avons simplifié en retirant les jonctions à puissance nulle. Nous observons bien que la résistance R et l'inductance L (associées en série) sont reliées à travers une jonction 1, permettant de symboliser dans le domaine électrique une association série.

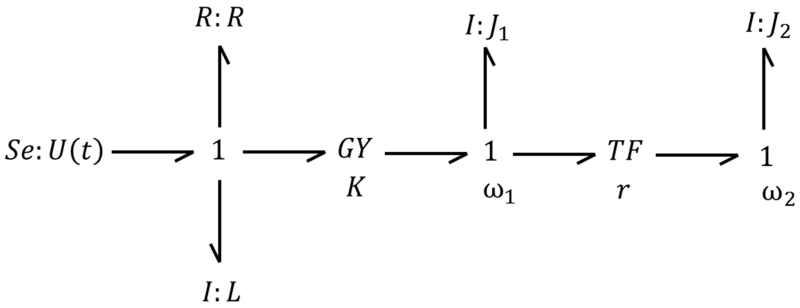


Figure 61. Bond graph acausal du moteur associé au réducteur

6.4 La causalité

La causalité permet d'exprimer un lien de cause à effet. Elle apparaît sous la forme d'un trait perpendiculaire au lien. La Figure 62 illustre les deux cas possibles. Le trait causal est placé du côté où l'effort est une donnée. Sur le cas de gauche, nous voyons que *A* impose un effort à *B* qui réagit en envoyant un flux à *A*. Le trait causal est donc placé du côté de *B*. Dans le cas de droite, *A* envoie un flux à *B* qui réagit par un effort. Le trait causal est donc placé du côté de *A*. Le signe « = » est symétrique, or pour traduire la causalité en équation, il nous faut introduire le signe « := » bien connu en algorithmique et en informatique. Ce signe permet d'exprimer clairement, dans le cas de l'exemple de gauche, que l'effort e_B ne peut être déterminé qu'après l'évaluation de e_A.

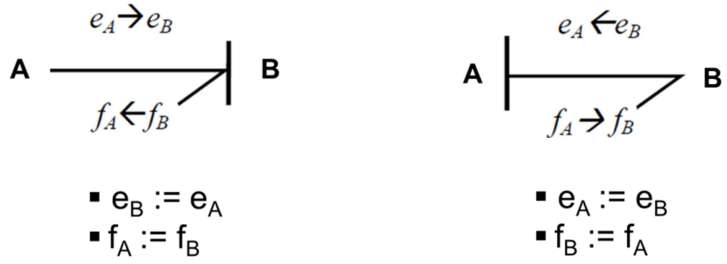


Figure 62. Les deux cas possibles de causalité

La causalité sur un Bond graph se déduit à partir de certaines règles qui sont présentées dans le Tableau 24. Pour les sources de flux et d'effort, la causalité est imposée. La première étape à effectuer pour déterminer la causalité d'un bond graph consiste donc à fixer la causalité des sources et de la propager sur l'environnement en respectant les contraintes fixées par les jonctions 0 et 1 ainsi que par les transformateurs et les gyrateurs. Ensuite, il faut de préférence affecter aux éléments *C* et *I* la causalité de type intégrale puis répercuter cela sur l'environnement. La causalité sur les éléments R peut, elle, être affectée de façon arbitraire. En cas de conflit, il faut revenir en arrière : rechercher l'élément C ou I qui en est la cause et le mettre en causalité dérivée. Les éléments de type R, dont l'affectation de la causalité est arbitraire introduisent des degrés de liberté dans la procédure d'affectation : il peut donc y avoir différentes possibilités de causalité pour un bond graph. L'organigramme de la Figure 63 présente la procédure d'affectation de la causalité.

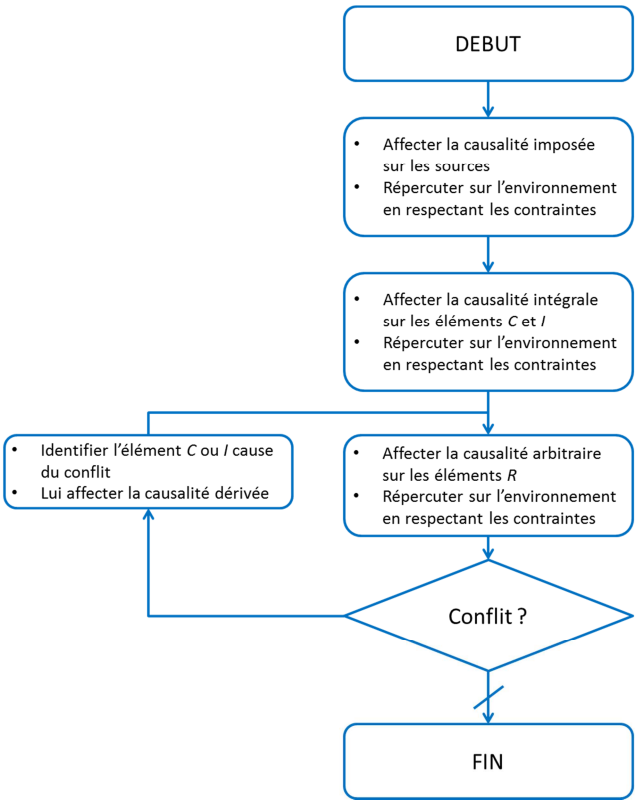
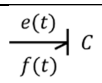
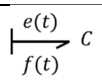
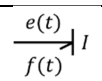
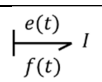
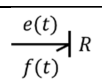
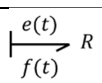
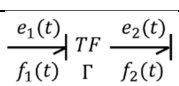
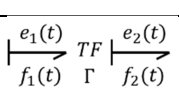
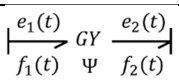
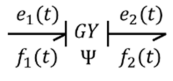
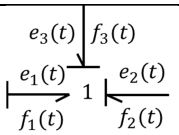
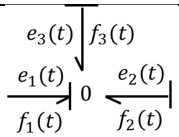


Figure 63. Procédure d'affectation de la causalité

Tableau 24. Les différents types de causalité

Représentation	Equation correspondante	Type de causalité
	$f(t) \coloneqq \frac{d}{dt}[\Phi_C . e(t)]$	Dérivée
	$e(t) \coloneqq {\Phi_C}^{-1} . q(t)$	Intégrale
	$f(t) \coloneqq {\Phi_I}^{-1} . p(t)$	Intégrale
	$e(t) \coloneqq \frac{d}{dt}[\Phi_I . f(t)]$	Dérivée
	$f(t) \coloneqq {\Phi_R}^{-1} . e(t)$	Conductance
	$e(t) \coloneqq \Phi_R . f(t)$	Résistance
	$e(t) \coloneqq E(t)$	Imposée
	$f(t) \coloneqq F(t)$	Imposée
	$\begin{cases} e_2(t) \coloneqq \frac{1}{\Gamma} . e_1(t) \\ f_1(t) \coloneqq \frac{1}{\Gamma} . f_2(t) \end{cases}$	Inverse
	$\begin{cases} e_1(t) \coloneqq \Gamma . e_2(t) \\ f_2(t) \coloneqq \Gamma . f_1(t) \end{cases}$	Directe
	$\begin{cases} e_1(t) \coloneqq \Psi . f_2(t) \\ e_2(t) \coloneqq \Psi . f_1(t) \end{cases}$	Directe
	$f_1(t) \coloneqq \frac{1}{\Psi} . e_2(t)$ $f_2(t) \coloneqq \frac{1}{\Psi} . e_1(t)$	Inverse
	$\begin{cases} f_2(t) \coloneqq f_1(t) \\ f_3(t) \coloneqq f_1(t) \\ e_1(t) \coloneqq -e_2(t) - e_3(t) \end{cases}$	Un seul flux imposé sur une jonction 1
	$\begin{cases} e_2(t) \coloneqq e_1(t) \\ e_3(t) \coloneqq e_1(t) \\ f_1(t) \coloneqq -f_2(t) - f_3(t) \end{cases}$	Un seul effort imposé sur une jonction 0

Le bond graph causal de l'exemple du moteur associé au réducteur est donné par la Figure 64. Tout d'abord, la source d'effort voit sa causalité imposée. Aucun autre trait causal ne peut être déduit de cela. Ensuite, Nous affectons la causalité intégrale aux éléments *I*, soit *L*, J_1 et J_2. De cela, nous pouvons déduire la causalité sur la liaison 1 associée à la vitesse ω_2 et donc sur le transformateur (causalité inverse) puis sur la liaison 1 associée à la vitesse ω_1 et enfin sur le gyrateur (causalité directe). Afin de ne pas créer de conflit au niveau de la liaison 1 correspondant au domaine électrique, l'élément *R* doit être affecté en causalité résistance.

Il apparaît un conflit au niveau de la liaison 1 correspondant à la vitesse ω_1 : deux flux y sont imposés, ce qui constitue une situation non conforme. L'élément en cause est toutefois facilement identifiable : il s'agit de l'élément $I : J_1$. Nous lui affectons donc la causalité dérivée et aucune autre modification n'est nécessaire.

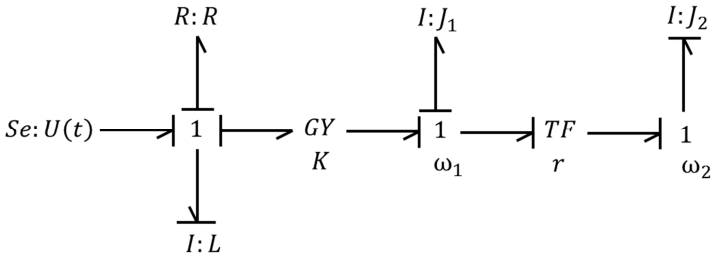


Figure 64. Bond graph causal du moteur associé au réducteur

6.5 Déduction du modèle mathématique et du schéma-bloc

Il est possible de déterminer le modèle mathématique correspondant à un bond graph. Il faut pour cela disposer du bond graph causal afin de pouvoir écrire les équations caractérisant le système. En effet, suivant la causalité qui est affectée à un élément, l'équation qu'il introduit diffère (cf. Tableau 24). Prenons l'exemple du circuit de la Figure 65, en appliquant les méthodes systématiques décrites précédemment, nous obtenons le bond graph simplifié puis le bond graph causal qui lui correspond et qui est donné par la Figure 66**.** A partir du bond graph causal, nous pouvons écrire les équations qui régissent le comportement du circuit. Ces équations sont présentées dans le Tableau 25.

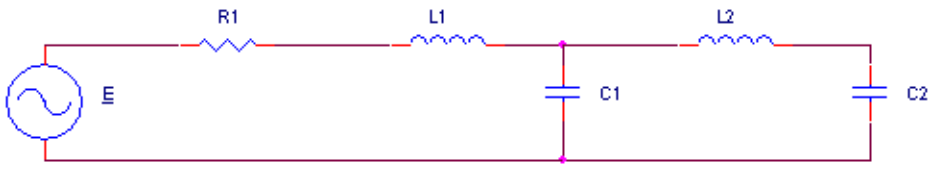


Figure 65. Exemple de circuit

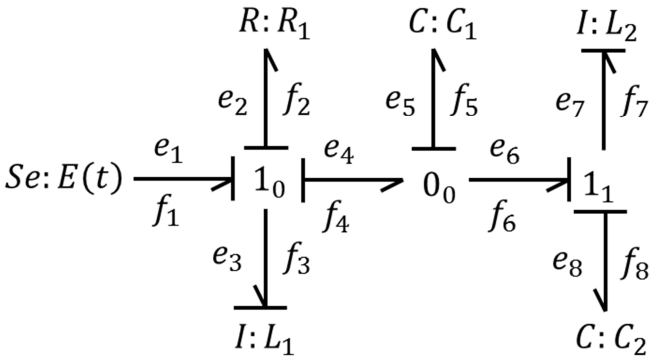


Figure 66. Bond graph simplifié de l'exemple

Tableau 25. Equations écrites à partir du bond graph causal de l'exemple

Elément	Equation correspondante	Type de causalité
E	$e_1 = E$	Imposée
R_1	$e_2 = R_1.f_2$	Résistance
L_1	$f_3 = \frac{1}{L_1}.p_3 = \frac{1}{L_1}.\int e_3.dt$	Intégrale
C_1	$e_5 = \frac{1}{C_1}.q_5 = \frac{1}{C_1}.\int f_5.dt$	Intégrale
C_2	$e_8 = \frac{1}{C_2}.q_8 = \frac{1}{C_2}.\int f_8.dt$	Intégrale
L_2	$f_7 = \frac{1}{L_2}.p_7 = \frac{1}{L_2}.\int e_7.dt$	Intégrale
Liaison 1_0	$\begin{cases} e_3 = e_1 - e_2 - e_4 \\ f_1 = f_3 \\ f_2 = f_3 \\ f_4 = f_3 \end{cases}$	
Liaison 0_0	$\begin{cases} f_5 = f_4 - f_6 \\ e_4 = e_5 \\ e_6 = e_5 \end{cases}$	
Liaison 1_1	$\begin{cases} e_7 = e_6 - e_8 \\ f_6 = f_7 \\ f_8 = f_7 \end{cases}$	

Ces équations permettent de déterminer le modèle mathématique correspondant au bond graph. Une façon d'écrire l'équation d'état est donnée par l'équation (74). Si l'on définit la sortie du système comme étant la tension aux bornes du condensateur C_2, alors l'équation de sortie est donnée par l'équation (75).

$$\begin{pmatrix} \dot{f}_3 \\ \dot{f}_7 \\ \dot{e}_5 \\ \dot{e}_8 \end{pmatrix} = \begin{pmatrix} -R_1/L_1 & 0 & -1/L_1 & 0 \\ 0 & 0 & 1/L_2 & -1/L_2 \\ 1/C_1 & -1/C_1 & 0 & 0 \\ 0 & 1/C_2 & 0 & 0 \end{pmatrix} . \begin{pmatrix} f_3 \\ f_7 \\ e_5 \\ e_8 \end{pmatrix} + \begin{pmatrix} 1/L_1 \\ 0 \\ 0 \\ 0 \end{pmatrix} . E \tag{74}$$

$$e_8 = (0 \quad 0 \quad 0 \quad 1) . \begin{pmatrix} f_3 \\ f_7 \\ e_5 \\ e_8 \end{pmatrix} \tag{75}$$

Les équations du Tableau 25 permettent aussi de construire facilement le schéma-bloc associé qui est présenté sur la Figure 67. Tous les efforts apparaissent en rouge et tous les flux apparaissent en vert. Le moment p_3 associé à l'élément $I:L_1$, le moment p_7 associé à l'élément $I:L_2$, le déplacement q_5 associé à l'élément $C:C_1$ et le déplacement q_8 associé à l'élément $C:C_2$ apparaissent en bleu. Ce schéma-bloc obtenu est un modèle plus important au niveau du nombre d'équations que les schémas-blocs que l'on a l'habitude de manipuler à haut-niveau d'abstraction. En effet, contrairement aux modèles comportementaux de systèmes linéaires, qui sont typiquement modélisés en SystemC AMS en utilisant le MoC LSF, nous avons ici affaire à un schéma-bloc conservatif en puissance.

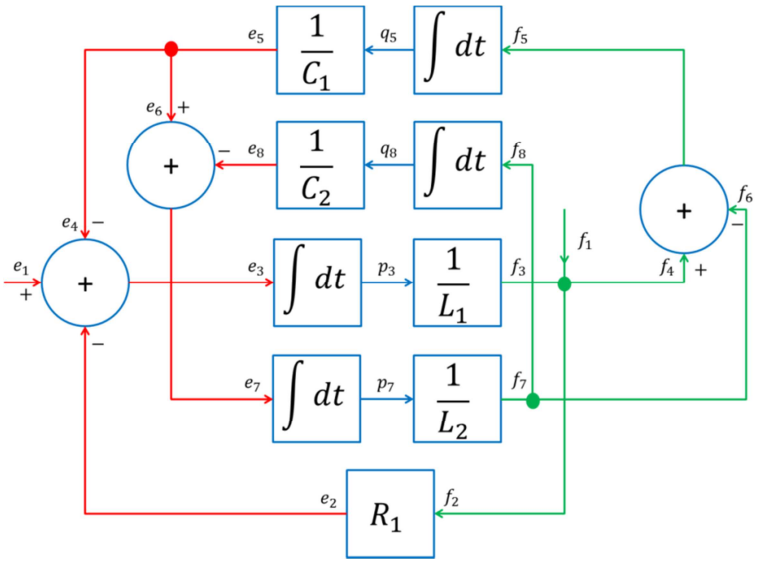


Figure 67. Schéma-bloc conservatif de l'exemple

Les trois représentations : le circuit électrique, le bond graph causal et le schéma-bloc conservatif contiennent autant d'informations. Cependant, par rapport au circuit, le bond graph causal permet d'abstraire le domaine physique : nous ne pensons plus en termes de tension et de courant, de force et de vitesse ou encore de pression et de débit mais en termes d'effort et de flux. Le schéma-bloc permet quant à lui d'abstraire l'effort et le flux en signaux dirigés. Ainsi, le schéma-bloc conservatif apparaît comme étant la représentation la plus abstraite des trois, le circuit électrique étant la moins abstraite et le bond graph causal se situant à un niveau intermédiaire.

6.6 Génération d'un modèle bond graph à partir d'un modèle bas niveau

Nous allons maintenant présenter une méthode pemrettant de générer automatiquement un bond graph à partir d'un modèle de bas niveau.

6.6.1 Présentation du modèle final

Il nous faut d'abord définir ce que nous voulons obtenir. Bien sûr, le but final qui apparaît comme étant le plus logique serait d'instancier automatiquement un bond graph sous SystemC AMS en utilisant les éléments de la bibliothèque SCAX [Mähne 11]. Cette solution semble toutefois quelque peu restrictive : obtenir une représentation équivalente au bond graph mais qui pourrait servir de base à la construction d'un modèle au moyen d'un autre outil que SystemC AMS semble plus pertinent.

Nous allons donc définir une représentation basée sur des tableaux (ou matrices) qui est équivalente aux bond graphs. Ce modèle constituera le modèle final que nous souhaitons obtenir à l'issue de l'exécution de la procédure automatique de génération du bond graph. Cette représentation est constituée de deux tableaux à deux dimensions, mais qui utilise en pratique le type SystemC AMS `sca_util::sca_matrix` qui permet de simplifier certaines manipulations. Nous utiliserons par la suite le terme « matrice » plutôt que celui de « tableau ». Il est aussi nécessaire de stocker le type et la valeur de chaque élément. Une première matrice doit pouvoir rendre compte du transfert d'énergie entre les jonctions : nous avons donc besoin d'une matrice carrée dont les lignes et les colonnes correspondent aux jonctions du bond graph. Nous appelons cette matrice la matrice de jonctions *J*. Elle nous permet de faire apparaître les jonctions qui sont connectées entre elles et celles qui ne le sont pas. Si la jonction d'indice *i* est reliée à la jonction d'indice *j* alors l'élément J_{ij} de la matrice est non nul (il prend alors la valeur « 1 »), sinon il est nul (« 0 »). Dans le cas ou $i = j$, alors l'élément J_{ij} est nul. Cette matrice doit aussi pouvoir rendre compte du sens du transfert de la puissance (le sens de la demi-flèche sur le bond graph). Pour faire apparaître cela, nous utilisons le signe. Si le transfert de puissance s'effectue de la jonction d'indice *i* vers la jonction d'indice *j* alors la valeur de l'élément J_{ij} sera positive et donc inchangée (« 1 ») et celle de l'élément *Jji* sera modifiée pour être négative (« -1 »). Le dernier concept qu'il est nécessaire de faire apparaître est la causalité. La causalité étant représentée par un trait sur un bond graph, nous avons opté pour ajouter un « 1 » à la valeur de l'élément concerné. Ainsi, si

la jonction d'indice *i* impose un effort à la jonction d'indice *j* et que le sens du transfert de puissance s'effectue dans le sens de *i* vers *j*, alors l'élément J_{ij} prendra pour valeur « 11 » et l'élément J_{ji} aura pour valeur « -1 ». Nous présentons ci-dessous la matrice de jonctions du bond graph de la Figure 66. Les lignes correspondent de haut en bas aux jonctions 1_0, 0_1 et 1_1. Les colonnes respectent le même ordre de gauche à droite.

$$J = \begin{pmatrix} 0 & 11 & 0 \\ -1 & 0 & 1 \\ 0 & -11 & 0 \end{pmatrix}$$

Une seconde matrice doit permettre de faire apparaître quels éléments sont connectés à telle ou telle jonction, et comment ils y sont connectés. Nous appelons cette matrice la matrice des éléments *E*. Ses lignes correspondent aux jonctions du bond graph alors que ses colonnes correspondent aux éléments. De même que pour la matrice de jonctions, si un élément d'indice *j* est relié à une jonction d'indice *i* alors la valeur de E_{ij} est non nulle (« 1 »), sinon la valeur de E_{ij} est nulle (« 0 »). Ensuite, si le sens du transfert de la puissance s'effectue de la jonction d'indice *i* vers l'élément d'indice *j* alors la valeur de E_{ij} demeure égale à « 1 », sinon, si le transfert de puissance s'effectue de l'élément d'indice *j* vers la jonction d'indice *i* alors E_{ij} est modifié et prend la valeur « -1 ». Enfin, dans le cas où la jonction d'indice *i* impose un effort à l'élément d'indice *j* et que le transfert de puissance s'effectue de la jonction vers l'élément alors la valeur de E_{ij} devient « 11 ». La matrice ci-dessous correspond à la matrice des éléments du bond graph de la Figure 66. Les lignes correspondent de haut en bas aux jonctions 1_0, 0_1 et 1_1. Les colonnes correspondent de gauche à droite aux éléments *E*, R_1, L_1, C_1, C_2 et L_2.

$$E = \begin{pmatrix} -11 & 11 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 11 & 1 \end{pmatrix}$$

Il apparaît que les deux matrices peuvent être fusionnées afin de n'en former plus qu'une seule, que nous appelons la matrice du bond graph *BG* et qui est obtenue en concaténant la matrice *E* à la matrice *J*. En effet, ces deux matrices ont le même nombre de lignes (le nombre de jonctions du bond graph), et cela permet de ne manipuler qu'une seule matrice. Nous donnons ci-dessous la matrice BG du bond graph de la Figure 66. Cette représentation est équivalente au bond graph comme nous allons le démontrer.

$$BG = \begin{pmatrix} 0 & 11 & 0 & -11 & 11 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -11 & 0 & 0 & 0 & 0 & 0 & 11 & 1 \end{pmatrix}$$

Supposons que l'on dispose de la matrice *BG*, est-on capable de construire le bond graph associé ? Pour la suite, les lignes de la matrice *BG* seront appelées X_0, X_1 et X_2 et les colonnes seront appelées Y_0, Y_1, Y_2, Y_3, Y_4 et Y_5. Tout d'abord, la matrice *BG* contient trois lignes, cela veut dire que le bond graph associé est constitué de trois jonctions. Les trois premières colonnes correspondent aussi aux jonctions. Etant donné que la matrice contient neuf colonnes, nous savons que le bond graph contient six éléments.

La première ligne de la matrice *BG* permet de savoir que :

- La jonction X_0 impose un flux à la jonction X_1 et que le transfert de puissance s'effectue de X_0 vers X_1.
- Un élément Y_0 impose un effort à la jonction X_0 et que le transfert de puissance s'effectue de Y_0 vers X_0.
- La jonction X_0 impose un flux à un élément Y_1 et que le transfert de puissance s'effectue de X_0 vers Y_1.
- Un élément Y_2 impose un flux à la jonction X_0 et que le sens du transfert de puissance s'effectue de X_0 vers Y_2.

Ces informations permettent de construire la structure autour de la jonction X_0. Nous donnons cette structure sur la Figure 68.

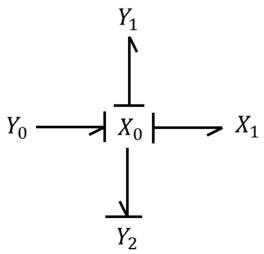


Figure 68. Bond graph déduit de la première ligne de la matrice *BG*

La seconde ligne de la matrice *BG* permet de savoir que :

- La jonction X_1 impose un effort à la jonction X_0 et que le transfert de puissance s'effectue de X_0 vers X_1.
- La jonction X_1 impose un effort à la jonction X_2 et que le transfert de puissance s'effectue de X_1 vers X_2.

- Un élément Y_3 impose un effort à la jonction X_1 et que le transfert de puissance s'effectue de X_1 vers Y_3.

Ces informations permettent de construire la structure autour de la jonction X_1. Nous donnons cette structure sur la Figure 69.

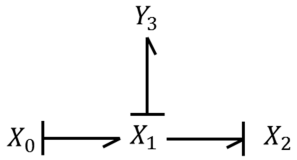


Figure 69. Bond graph déduit de la seconde ligne de la matrice *BG*

La troisième ligne de la matrice *BG* permet de savoir que :

- La jonction X_2 impose un flux à la jonction X_1 et que le transfert de puissance s'effectue de X_1 vers X_2.
- Un élément Y_4 impose un effort à la jonction X_2 et que le transfert de puissance s'effectue de X_2 vers Y_4.
- Un élément Y_5 impose un flux à la jonction X_2 et que le transfert de puissance s'effectue de X_2 vers Y_5.

Ces informations permettent de construire la structure autour de la jonction X_2. Nous donnons cette structure sur la Figure 70.

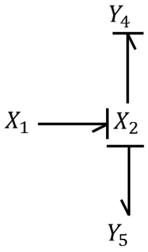


Figure 70. Bond graph déduit de la troisième ligne de la matrice *BG*

Les trois structures permettent de construire le bond graph global, qui est donné par la Figure 71. Les natures de chacune des structures peuvent être déduites. Nous savons qu'un seul flux peut être imposé sur une liaison 1, cela implique que X_0 et X_2 sont des liaisons 1. D'autre part, nous savons qu'un seul effort peut être imposé sur une liaison 0, cela implique alors que X_1 est une liaison 0. Cependant, il est plus pratique et plus sûr (impossible de déterminer la type d'une jonction qui n'a que deux liens) d'associer deux tableaux de chaines de caractères à

cette matrice. Un premier tableau correspondant aux noms des jonctions (lignes) et un second tableau correspondant aux noms des éléments. A partir de la matrice *BG*, nous obtenons donc bien un modèle équivalent à celui de la Figure 66.

.

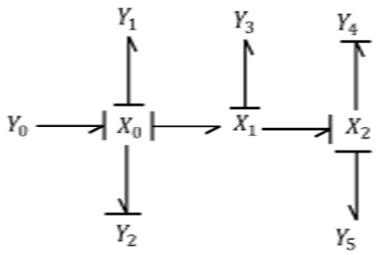


Figure 71. Bond graph global déduit à partir de la matrice *BG*

6.6.2 Présentation du modèle de départ

Nous venons de définir le modèle auquel nous voulons que notre méthode aboutisse. Il nous faut désormais définir le modèle de départ. La difficulté réside dans le fait que les systèmes que nous souhaitons modéliser sont électromécaniques. La netlist est un bon modèle de bas niveau pour le domaine électrique, de plus nous disposons d'outils permettant d'analyser les netlist. Ainsi apparaît l'idée d'étendre ce mode de représentation afin de pouvoir y intégrer des éléments mécaniques, la netlist devenant électromécanique. Une telle approche a déjà été utilisée dans [Clark 07] pour les MEMS, cependant ce travail était focalisé sur les systèmes à paramètres distribués. Notre travail est porté sur les macro systèmes : il nous faut donc intégrer le ressort (K), la masse (M) et l'amortisseur (D) ainsi que la force (F) et la vitesse (U). La syntaxe demeure la même que pour une netlist électrique : nom de l'élément, liste des nœuds (ou points) auxquels l'élément est connecté, valeur de l'élément. L'exemple ci-dessous montre le cas d'un système masse-ressort-amortisseur, voir Figure 43. Nous notons que la masse M1 n'est connectée qu'à un seul « nœud ». En effet, pour des questions de commodités, au niveau du développement du programme, nous considérons que toute la masse d'un objet est concentrée en un seul point. Cette approche est similaire à celle de la mécanique du point où l'on étudie le mouvement du centre d'inertie de l'objet.

```
K1 0 1 K1
D1 0 1 D1
M1 1 M1
```

Si l'on se réfère à l'algorigramme de la Figure 58, on voit que la première étape consiste à identifier les domaines physiques. Ceci peut être fait aisément à la simple lecture de la première lettre d'une ligne de la netlist. En effet, les éléments K, D, M, F et U désignent des éléments mécaniques alors que les éléments C, R, L, V et I désignent des éléments

électriques. Il est important de signaler ici qu'un moteur ne fait pas partie de la liste des composants que notre méthode est capable de traiter. L'utilisateur devra donc spécifier dans la netlist son modèle équivalent : résistance, inductance et gyrateur. Ainsi, la netlist ci-dessous présente le modèle de bas niveau correspondant au système motoréducteur de la Figure 59 C'est ce type de modèle qui constitue le point d'entrée de notre méthode.

```
V1 1 0 V1
R1 1 2 R1
L1 2 3 L1
GY 3 0 4 10 Psi
M1 4 M1
TF 4 10 5 20 Ratio
M2 5 M2
```

6.6.3 Présentation de la méthode de génération automatique de bond graph

Nous avons défini le modèle de bas niveau que nous allons utiliser afin de générer automatiquement la matrice *BG* équivalente au bond graph. L'algorigramme de la Figure 58 indique que la première étape consiste à identifier les domaines physiques. Nous pouvons donc séparer la netlist électromécanique en une netlist électrique et une netlist mécanique, les éléments de couplage apparaissant dans les deux netlist. Dans le cas du motoréducteur, la netlist de la partie électrique correspond à :

```
V1 1 0 V1
R1 1 2 R1
L1 2 3 L1
GY1 3 0 4 10 Psi
```

La netlist de la partie mécanique correspond quant à elle à:

```
GY1 3 0 4 10 Psi
M1 4 M1
TF 4 10 5 20 Ratio
M2 5 M2
```

Il faut préciser ici que les éléments de couplage que sont le transformateur et le gyrateur ne doivent pas être traités comme les autres éléments. En effet, sur un bond graph, un élément (*R*, *C* ou *I*) est considéré comme un élément à un port : il ne peut être connecté qu'à une seule jonction. Les gyrateurs et les transformateurs doivent être considérés comme des éléments à deux ports. En pratique, lorsque ce composant est utilisé pour réaliser l'interface électromécanique nous pouvons le considérer comme un élément à un port. Il faudra alors le prendre en compte à la fois dans l'analyse de la partie électrique et dans l'analyse de la partie mécanique. Dans chacune de ces analyses, il sera considéré comme un élément à un port.

C'est l'étape de fusion du bond graph électrique et du bond graph mécanique qui nous permettra de le considérer comme un élément à deux ports.

Si la fonction du transformateur ou du gyrateur n'est pas de réaliser l'interface électromécanique, il est plus simple de l'assimiler à une liaison.

Il faut maintenant effectuer une analyse qui diffère suivant le domaine physique. Nous allons d'abord générer le bond graph de la partie électrique. Nous verrons ensuite comment générer le bond graph de la partie mécanique.

6.6.3.1 Bond graph de la partie électrique

La première étape de la méthode permettant d'obtenir le bond graph de la partie électrique consiste à créer une jonction 0 pour chaque potentiel électrique. La seconde étape consiste à ajouter une jonction 1 pour chaque différence de potentiel induite par un composant, soit une jonction 1 par composant. La troisième étape consiste à ajouter une jonction 0 par jonction 1 créée précédemment. Ces trois étapes peuvent être rassemblées en une seule consistant à créer toutes les jonctions de la partie électrique.

Le nombre total de jonctions à créer est donc égal à *n+2c*, avec *n* le nombre de potentiels et *c* le nombre de composants. L'idée est donc de créer une matrice (type SystemC AMS : `sca_util::sca_matrix`) carrée qui nous permette de faire apparaître la façon dont les liaisons sont connectées entre elles. Cette matrice a la même fonction que la matrice *J* dont nous avons parlé en 6.6.1. Mais ici elle ne concerne que le bond graph de la partie électrique. Nous créons donc une matrice dont les lignes et les colonnes correspondent aux jonctions. Cette matrice de jonctions contient un très grand nombre de « 0 » : en effet, un « 0 » apparaît sur l'élément d'indice *ij* si une liaison d'indice *i* n'est pas connectée à la liaison d'indice *j*. Un tableau de chaines de caractères est associé à la matrice. Le nom de chaque jonction y est entré. Cela nous permet d'identifier aisément les éléments de la matrice qui sont non nuls. Nous avons deux types de liaisons : les liaisons 0 et les liaisons 1. Les liaisons 1 correspondent à des différences de potentiels. Les liaisons 0 sont séparées en deux sous-types : celles qui se trouvent placées entre deux liaisons 1 (qui correspondent à un potentiel) et celles qui se trouvent placées entre une liaison 1 et un élément (qui correspondent à une différence de potentiel). Nous avons donc des liaisons de type 0_X, 1_{YZ} et 0_{ST}. Les liaisons 0_X ne peuvent – à ce stade – être connectées qu'à des liaisons de type 1_{YZ}. Elles le seront si $y = x$ ou si $z = x$. Les liaisons de type 0_{ST} ne peuvent quant à elles être connectées qu'à des liaisons de type 1_{YZ}. Elles le seront si $s = y$ et $t = z$. Dans le cas du motoréducteur, nous obtenons une matrice 12*12 que nous ne montrerons pas ici. Cette matrice permet de faire apparaître les liens qui existent entre les différentes jonctions. Le sens du transfert de puissance ainsi que la causalité ne peuvent être déterminés qu'en fonction des éléments.

Il nous faut donc construire la matrice des éléments. Au stade où nous en sommes, un élément ne peut être connecté qu'à une jonction de type 0_{ST}, cela implique que cette matrice aura *c*

lignes et *c* colonnes, avec *c* le nombre de composants. Les lignes de cette matrice correspondent aux jonctions de types 0_{ST}, les colonnes correspondent aux composants. Il est ici très facile à identifier les éléments non nuls. Chaque ligne correspond à une différence de potentiel introduite par un composant, il s'agit donc de déterminer quel est le composant qui introduit ladite différence de potentiel. Cela est fait en identifiant les nœuds auxquels il est connecté. Dans cette matrice, nous pouvons aussi faire apparaître le sens du transfert de puissance : si la jonction est connectée à un composant passif, il faut écrire un « 1 », si elle est connectée à une source, il faut écrire un « -1 ». Nous montrons ci-dessous la matrice des éléments E_E qui correspond à la partie électrique du système moteur-réducteur. Les lignes correspondent de haut en bas aux jonctions 0_{12}, 0_{23}, 0_{30}, 0_{10}. Les colonnes correspondent de gauche à droite aux composants V_1, R_1, L_1 et GY_1 (traité ici comme un élément à un port puisqu'il est un élément de couplage électromécanique).

$$E_E = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \end{pmatrix}$$

Le sens du transfert de la puissance entre les composants et les jonctions de type 0_{ST} étant désormais connu, il doit être propagé aux autres types de jonctions. Il faut tout d'abord propager le sens du transfert de puissance des jonctions 0_{ST} vers les jonctions 1_{YZ} : le sens du transfert est le même qu'entre les composants et les jonctions 0_{ST}. Si le transfert de puissance se fait d'une liaison 0_{ST} vers une liaison 1_{YZ} alors l'élément [0_{ST} ; 1_{YZ}] de la matrice aura pour valeur « 1 » et l'élément [1_{YZ} ; 0_{ST}] de la matrice aura pour valeur « -1 ». Il s'agit maintenant de propager le sens du transfert de puissances entre les liaisons 1_{ST} et les liaisons 0_X. A ce stade, les seules liaisons 1_{YZ} qui reçoivent de la puissance sont celles qui sont reliées, *via* une liaison 0_{ST} à une source. Ces liaisons 1_{YZ} fournissent donc nécessairement de la puissance : elles ne peuvent pas recevoir de la puissance provenant d'une autre source. D'autre part, nous savons que le potentiel de référence ne peut qu'être récepteur de puissance. De plus, les jonctions de types 0_X n'étant à ce stade reliées qu'à deux liaisons de type 1_{YZ}, mis à part la liaison 0_X associée au potentiel de référence, elles reçoivent de la puissance d'une part et transmettent la puissance d'autre part. Ces trois aspects nous permettent de propager le sens du transfert de puissance et de compléter la matrice de jonction. La dernière étape consiste à supprimer les jonctions qui correspondent aux potentiels de référence ainsi que les liens qui y sont attachés. Cette dernière opération est effectuée en identifiant l'indice *i* de la jonction 0_X correspondant au nœud de référence : soit la jonction 0_0. Ensuite, il s'agit simplement de supprimer la ligne et la colonne qui portent le même indice *i*. La matrice J_E ci-dessous est la matrice de jonctions de la partie électrique du système moteur-réducteur. Les lignes et les colonnes de cette matrice correspondent de haut en bas et de gauche à droite aux jonctions : 0_1, 0_2, 0_3, 1_{12}, 1_{23}, 1_{30}, 1_{10}, 0_{12}, 0_{23}, 0_{30} et 0_{10}. Nous ne pouvons pas fusionner les matrices J_E et E_E pour l'instant : la matrice J_E contient onze lignes alors que la matrice E_E n'en contient que

quatre. Ce n'est qu'après l'étape de simplification du bond graph que l'on peut fusionner les matrices.

$$J_E = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Enfin, jusqu'ici nous parlions de composant : la méthode bond graph utilise la dénomination « élément ». Ainsi, le type du composant électrique doit être modifié pour adopter la dénomination bond graph. Les matrices *E* et *J* nous permettent de construire le bond graph de la Figure 72. Ce bond graph est équivalent à la partie de gauche (correspondant à la partie électrique) de la Figure 60.

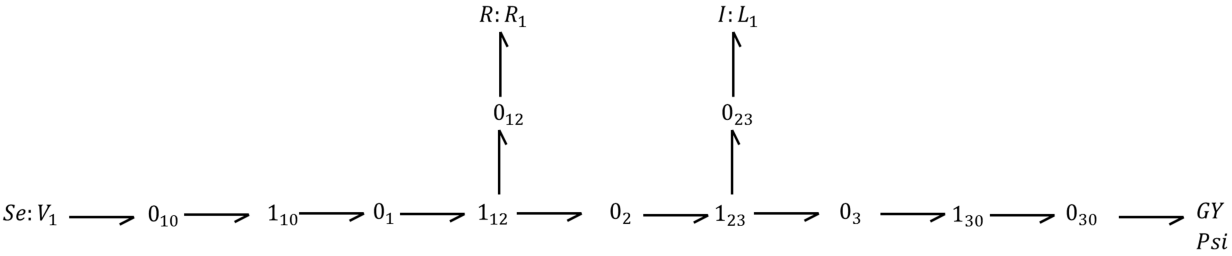


Figure 72. Bond graph de la partie électrique du système moteur-réducteur

6.6.3.2 Bond graph de la partie mécanique

La première étape de la construction du bond graph d'un système mécanique consiste à créer une jonction 1 pour chaque vitesse de composant. La seconde étape consiste à créer une jonction 0 pour chaque relation entre les vitesses. Ces deux étapes peuvent être regroupées en une seule qui consiste à créer toutes les jonctions. Les masses introduisent des vitesses absolues alors que les ressorts et les amortisseurs introduisent des vitesses relatives. Nous avons besoin de deux types de jonctions 1 pour faire apparaître cela : des jonctions de type 1_A et des jonctions de type 1_{BC}. Une jonction de type 1_A est associée à une vitesse absolue (vitesse d'une masse ou « source » de vitesse). Une jonction de type 1_{BC} est associée à la vitesse relative d'un ressort ou d'un amortisseur. Il existe un troisième type de jonctions : les jonctions de type 0_{DE}. Une jonction de type 0_{DE} est intercalée entre deux jonctions de type 1_A. Le cas du réducteur fait apparaître seulement deux vitesses absolues. Nous aurons donc deux

jonctions de type 1_A. Cependant, le transformateur permettant la réduction de la vitesse de rotation et donc l'augmentation du couple n'est pas ici un élément de couplage électromécanique : c'est un composant mécanique pur ayant deux ports mécaniques. Nous devons l'assimiler à une jonction, bien qu'aucun autre composant n'y soit lié. Le gyrateur, qui lui est un élément de couplage électromécanique peut être assimilé à un composant. La matrice de jonctions J_M du réducteur apparaît ci-dessous. Les lignes et les colonnes correspondent de haut en bas et de gauche à droite aux jonctions : 1_4, 1_5 et TF_{45}.

$$J_M = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

De même que pour la génération du bond graph de la partie électrique, il nous faut créer la matrice des éléments E_M afin de pouvoir par la suite déterminer le sens du transfert de puissance. La méthode est assez proche de celle permettant d'obtenir la matrice des éléments de la partie mécanique. Les composants ne peuvent à ce stade être connectés qu'aux jonctions de type 1_A ou de type 1_{BC}. Les colonnes de la matrice de jonction correspondent aux composants et les lignes correspondent aux jonctions. Une ligne correspondant à une jonction de type 1_A contiendra un « 1 » sur la colonne correspondant à une masse ou une « source » de vitesse connectée au « nœud » *A*. Une ligne correspondant à une jonction de type 1_{BC} contiendra un « 1 » sur la colonne correspondant à un ressort ou un amortisseur ou une force connectée entre les nœuds *B* et *C*. Le sens du transfert de puissance entraine une modification éventuelle d'un « 1 » à un « -1 », dans le cas où une source est connectée à une jonction. La matrice des éléments E_M du réducteur est donnée ci-dessous. Les lignes correspondent de haut en bas aux jonctions 1_4, 1_5 et TF_{45}. Les colonnes correspondent de gauche à droite aux composants GY_1, M_1 et M_2. Il n'y a aucune source à proprement parler dans ce système : c'est le gyrateur qui fournit la puissance

$$E_M = \begin{pmatrix} -1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

La dernière étape consiste à supprimer les jonctions 1 qui correspondent aux vitesses nulles ainsi que les liens qui y sont attachés. Cette dernière opération est effectuée en identifiant l'indice *i* de la jonction 1_A correspondant au nœud de référence : soit la jonction 1_0. Ensuite, il s'agit simplement de supprimer la ligne et la colonne qui portent le même indice *i*. Dans le cas du réducteur, il n'y a pas de vitesse nulle. Tout comme dans le cas de la partie électrique, le type des composants doit être modifié afin de se ramener à la dénomination bond graph. Nous remarquons qu'il serait possible ici de fusionner les matrices JM et EM, en effet, les lignes de ces deux matrices correspondent aux mêmes jonctions : 1_4, 1_5 et TF_{45}. Cela signifie que le bond graph de la partie mécanique qui apparaît sur la Figure 73 n'a pas besoin d'être simplifié.

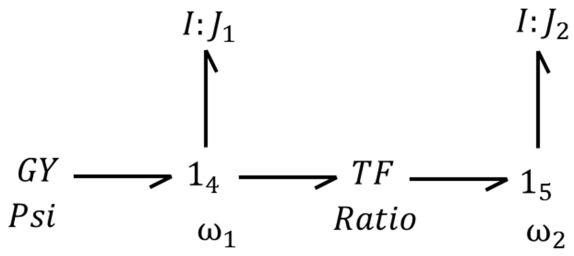


Figure 73. Bond graph de la partie mécanique du système moteur-réducteur

6.6.3.3 Fusion des bond graphs

Nous en arrivons donc à l'étape de fusion des bond graphs. Ce sont les éléments de couplage qui permettent de réaliser cela. Un élément de couplage a jusqu'ici été considéré comme un élément à un port dans le domaine électrique, de même pour le domaine mécanique. Nous allons maintenant le considérer comme un élément à deux ports : cela va nous permettre de faire le lien entre les deux domaines. La première étape consiste à fusionner les matrices des éléments E_E et E_M afin de créer la matrice E_{BG}. Cette matrice permet de faire apparaître comment les éléments sont liés aux jonctions, en intégrant le domaine électrique et le domaine mécanique. Il s'agit d'ajouter les lignes de la matrice J_M à la suite des lignes de la matrice J_E et d'ajouter les colonnes de la matrice E_M à la suite des colonnes de la matrice E_E. Après cela, l'élément de jonction fait doublon : une première colonne correspond à son interface électrique et une seconde colonne correspond à son interface mécanique. La dernière tâche à effectuer afin d'obtenir la matrice E_{BG} est donc de fusionner ces deux colonnes en une seule. La matrice des éléments du système moteur-réducteur E_{BG} est donnée ci-dessous. Les lignes de cette matrice correspondent de haut en bas aux jonctions : 0_{12}, 0_{23}, 0_{30}, 0_{10}, 1_4, 1_5 et TF_{45}. Les colonnes de cette matrice correspondent aux éléments : V_1, R_1, L_1, GY_1, M_1, et M_2.

$$E_{BG} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Il faut maintenant fusionner les matrices J_E et J_M. La procédure consiste ici simplement à ajouter les lignes de la matrice J_M à la suite de celles de la matrice J_E et d'ajouter les colonnes de la matrice J_M à la suite des colonnes de la matrice J_E afin d'obtenir la matrice des jonctions du bond graph J_{BG} qui est donnée ci-dessous. Les lignes et les colonnes de cette matrice correspondent de haut en bas et de gauche à droite aux jonctions du bond graph : 0_1, 0_2, 0_3, 1_{12}, 1_{23}, 1_{30}, 1_{10}, 0_{12}, 0_{23}, 0_{30}, 0_{10}, 1_4, 1_5 et TF_{45}.

$$J_{BG} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \end{pmatrix}$$

Jusqu'ici, pour des raisons de commodités, un élément de transformation qui ne permettait pas de coupler le domaine électrique avec le domaine mécanique était considéré comme une liaison, cependant, à ce stade, ce n'est plus pertinent. En effet, les éléments de type *R*, *C* ou *I* ne sont jamais reliés directement avec un transformateur ou un gyrateur, c'est pour cela que tous les éléments de la dernière ligne de la matrice E_{BG} ci-dessus sont nuls : aucun élément à un port ne peut y être connecté. L'idée est donc de « transformer » cette jonction en élément afin de n'avoir plus que des jonctions 0 ou 1 dans la liste des liaisons. Dans la matrice des éléments E_{BG}, cette opération se traduit pas la suppression d'une ligne et l'ajout d'une colonne. Dans la matrice des jonctions J_{BG}, cela se traduit par la suppression d'une ligne et d'une colonne. Toutefois, avant d'être supprimées, les éléments non nuls de ces lignes et colonnes sont récupérés afin de pouvoir compléter la colonne la matrice E_{BG} nouvellement créée. La nouvelle matrice E_{BG} du motoréducteur est donnée ci-après. Les lignes correspondent aux jonctions : 0_{12}, 0_{23}, 0_{30}, 0_{10}, 1_4 et 1_5. Les colonnes correspondent aux éléments V_1, R_1, L_1, GY_1, M_1, M_2 et TF_1.

$$E_{BG} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Nous ne montrons pas la nouvelle version de la matrice des jonctions J_{BG}. Par rapport à la version de la page précédente, il suffit de noter que la dernière ligne ainsi que la dernière colonne ont été supprimées. Il n'est cependant pas inutile de noter que les deux dernières lignes de cette matrice, qui correspondent aux jonctions 1_4 et 1_5 ne sont plus liées à aucune autre jonction. En effet, la jonction 1_4 est liée à l'élément gyrateur GY_1 et à l'élément transformateur TF_1 et la jonction 1_5 est seulement liée à l'élément transformateur TF_1. La Figure 74 présente le bond graph après la fusion. La prochaine étape consistera à le simplifier.

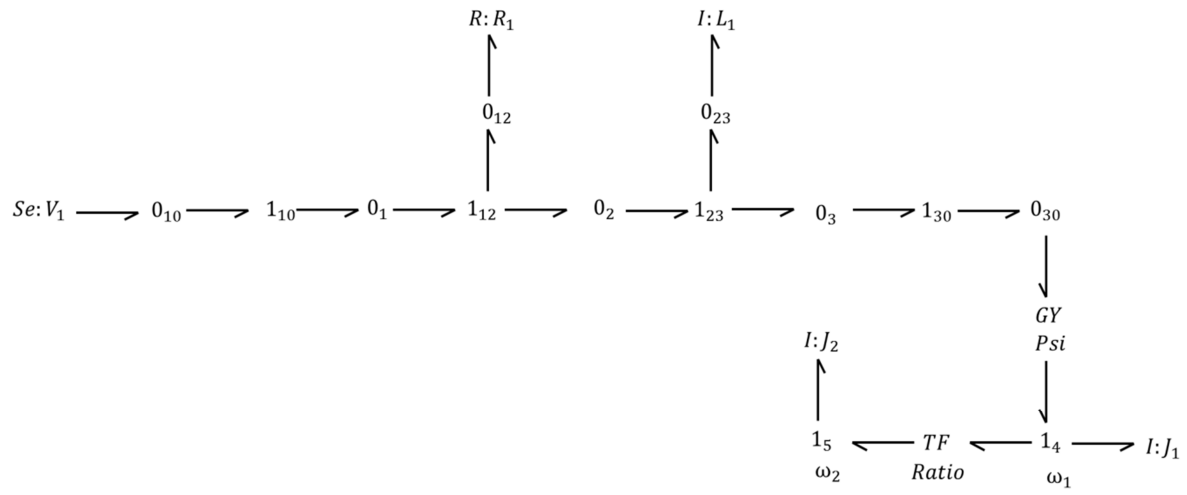


Figure 74. Bond graph non simplifié du système moteur-réducteur

6.6.3.4 Simplification du bond graph

Il s'agit ici de se ramener à la forme la plus simple du bond graph, il nous faut pour cela éliminer les jonctions superflues. La simplification se fait en deux étapes : la première consiste à ne garder qu'une seule jonction lorsque plusieurs jonctions transfèrent une puissance identique. La seconde étape consiste à ne garder qu'une seule jonction lorsque plusieurs jonctions du même type sont reliées.

La première observation que nous faisons est que lorsqu'une jonction *n* n'est reliée qu'à seulement deux jonctions, une jonction *n-1* qui lui fournit de la puissance et une jonction *n+1* à qui elle fournit de la puissance, alors les jonctions *n-1* et *n+1* peuvent être liée directement et la jonction *n* peut être supprimée puisque la puissance demeure constante. C'est ce qu'expriment clairement les deux premières lignes du Tableau 23. Une telle situation se traduit dans la matrice de jonctions par une ligne (et une colonne) qui contient deux éléments non nuls, l'un qui est égal à « -1 » et l'autre qui est égal à « 1 ». Prenons l'exemple de la première ligne la matrice J_{BG}, ligne qui correspond à la jonction 0_1. Nous voyons que cette jonction reçoit une puissance de la part de la jonction 1_{10}, et qu'elle la transmet intégralement à la jonction 1_{12}. La jonction 1_{10} peut donc être reliée directement à la jonction 1_{12} et la jonction 0_1 peut être supprimée. D'autres jonctions sont dans un cas identique : 1_{10}, 0_2, 0_3, 1_{30}. Il faut donc vérifier, pour chaque jonction, si la puissance qu'elle transmet est égale à la puissance qu'elle reçoit, si c'est le cas, elle peut être supprimée. La méthode consiste ici à suivre le sens du transfert de puissance en partant des jonctions connectées aux sources. A ce stade, une jonction 0_{YZ} qui est connectée à une source ne peut être connectée qu'à une seule autre jonction 1_{ST}, à qui elle fournit de l'énergie. Cela se traduit dans la matrice des jonctions par une ligne ou une colonne n'ayant qu'un seul élément non nul et égal à « 1 ». Cet élément non nul correspond à la jonction 0_{YZ} qui reçoit de la puissance de la part de la jonction connectée à la source. En récupérant l'indice de cet élément, on peut suivre le sens du transfert de puissance et vérifier si la jonction correspondante transmet de la puissance à une

seule autre jonction 0_X. Si c'est le cas, la jonction 0_{YZ} reliée à la source est directement reliée à cette jonction 0_X. Cette opération est réalisée très simplement par la modification des valeurs des éléments de la matrice qui sont concernés puis par la suppression de la ligne et de la colonne correspondantes à la jonction 1_{ST}. L'opération est répétée jusqu'à avoir traité toute la matrice de jonctions. Dans le cas du système motoréducteur, cela nous mène au bond graph de la Figure 75.

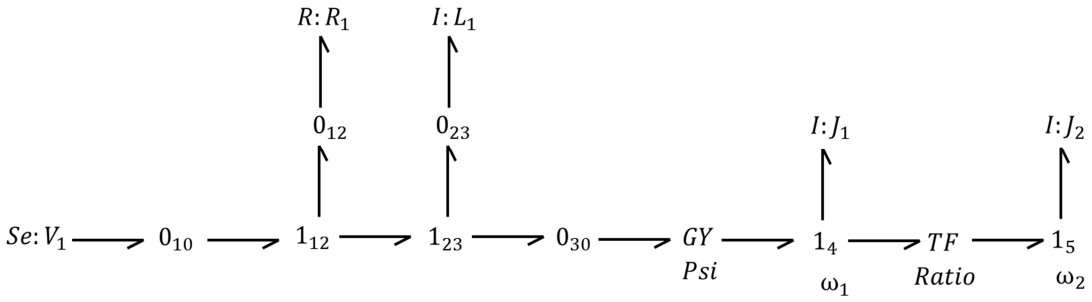


Figure 75. Bond graph du motoréducteur après la 1ere partie de la 1ere étape de la simplification

Cette étape n'est toutefois pas terminée, nous voyons sur la figure précédente que d'autres jonctions reçoivent puis transmettent des puissances équivalentes : ce sont des jonctions placées entre une autre jonction et un élément. 0_{10}, 0_{12}, 0_{23} et 0_{30} sont dans ce cas. Ces jonctions sont les jonctions qui permettent de relier les éléments. A ce stade, une jonction qui est connectée à un élément est obligatoirement aussi connectée à une seule autre jonction. Dans la matrice des jonctions, on identifie les lignes ou colonnes correspondantes à ces jonctions par la présence d'un seul élément non nul : «1 » pour une jonction qui est connectée à une source et qui fournit donc de la puissance à l'autre jonction à laquelle elle est connectée, « -1 » pour une jonction qui est connectée à un élément passif et qui reçoit donc de l'énergie de la part de l'autre jonction à laquelle elle est connectée. Ces lignes et colonnes peuvent être supprimées non sans avoir au préalable modifié en conséquence les éléments concernés de la matrice de jonction de la même façon que précédemment. La matrice des éléments n'est pas modifiée, mais le tableau qui lui est associé et qui permet d'identifier le nom des jonctions (lignes) est modifié. Le bond graph obtenu à la suite de cette opération est donné en Figure 76.

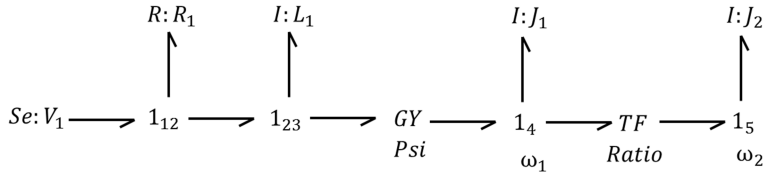


Figure 76. Bond graph du motoréducteur après la 2nde partie de la 1ere étape de la simplification

La seconde étape consiste maintenant à simplifier les structures reliées constituées de plusieurs jonctions du même type. Ce sont les cas se rapportant aux deux dernières lignes du Tableau 23. L'opération consiste ici à supprimer toutes les jonctions concernées sauf une, à laquelle on attache les éléments devenus « orphelins » (la jonction à laquelle l'élément était connecté a été supprimée). La méthode consiste ici à vérifier si des jonctions connectées entre elles ont un nom qui commence par le même caractère (nous rappelons qu'un tableau de chaines de caractères est associé à la matrice de jonctions). Si c'est le cas, alors toutes les jonctions sont supprimées sauf une. Il faut aussi modifier la matrice des éléments en conséquence. Le bond graph ainsi obtenu est donné par la Figure 77.

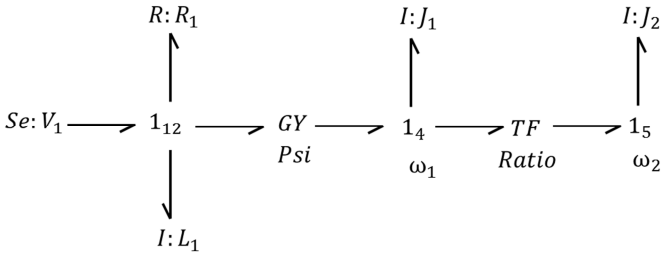


Figure 77. Bond graph du motoréducteur après la seconde étape de la simplification

La matrice des éléments E_{BG} du motoréducteur est donnée ci-dessous. Les lignes correspondent de haut en bas aux jonctions 1_{12}, 1_4 et 1_5. Les colonnes correspondent aux éléments V_1, R_1, L_1, GY_1, M_1, M_2 et TF_1.

$$E_{BG} = \begin{pmatrix} -1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

La matrice des jonctions J_{BG} du motoréducteur est donnée ci-dessous, les lignes et les colonnes correspondent de haut en bas et de gauche à droite aux jonctions 1_{12}, 1_4 et 1_5. La matrice ne contient que des « 0 » puisque nous n'avons aucun plus lien direct entre jonctions.

$$J_{BG} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Nous pouvons d'ores et déjà fusionner les matrices E_{BG} et J_{BG} afin de créer la matrice BG qui est donnée ci-dessous. Les lignes de cette matrice correspondent de haut en bas aux jonctions 1_{12}, 1_4 et 1_5. Les colonnes correspondent de gauche à droite aux jonctions 1_{12}, 1_4 et 1_5 puis aux éléments V_1, R_1, L_1, GY_1, M_1, M_2 et TF_1.

$$BG = \begin{pmatrix} 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

6.6.3.5 *Affectation de la causalité*

La dernière étape consiste à affecter la causalité aux jonctions et éléments du bond graph. Pour cela, il faut tout d'abord fixer les causalités imposées (sources). Une source d'effort permettra d'écrire « -11 » sur la ligne correspondant à la jonction à laquelle elle est connectée. Une source de flux permettra d'écrire « -1 » sur la ligne correspondant à la jonction à laquelle elle est connectée. La causalité peut éventuellement être propagée à partir des jonctions auxquelles sont reliées des sources, puisque nous savons, d'après le Tableau 23 qu'un seul flux peut être imposé sur une jonction 1 et qu'un seul effort peut être imposé sur une jonction 0. Cela signifie que sur une ligne correspondant à une jonction 1, il ne peut y avoir qu'un seul élément égal à « 1 » ou « -1 ». De même, sur une ligne correspondant à une jonction « 0 », il ne peut y avoir qu'un seul élément égal à « 11 » ou « -11 ».Ce sont ces deux règles qui nous permettent de vérifier qu'il n'y a pas de conflit. Ensuite, nous affectons dans un premier temps la causalité intégrale aux éléments de stockage *C* et *I*. Ceci permet de compléter la causalité de toutes les liaisons. Si une irrégularité au sens des deux règles de vérification se produit sur une jonction, alors il faut modifier la causalité d'un élément de stockage en le plaçant en causalité dérivée, puis propager la causalité. Si une irrégularité apparaît encore, il faut modifier la causalité d'un autre élément de stockage et ainsi de suite jusqu'à parvenir à déterminer un bond graph causal. Une boucle est donc effectuée. La Figure 78 montre le bond graph du motoréducteur obtenu au premier tour d'affectation de la causalité. Le trait causal de la liaison entre V_1 et 1_{12} apparaît en noir car il est définitif pour motif de causalité imposée. Les traits causaux apparaissant en bleu sont ceux qui sont dits préférentiels, dans un premier temps, on affecte la causalité intégrale aux éléments de stockage. Les traits causaux qui apparaissent en vert sont déduits. Ce bond graph n'est pas causal puisque la règle de vérification des jonctions 1 n'est pas respectée sur la jonction 1_5 : deux flux sont imposés sur cette jonction.

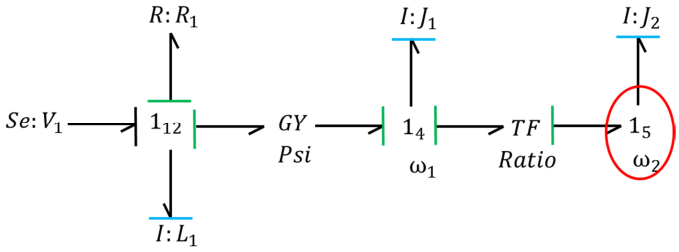


Figure 78. Bond graph non causal obtenu au premier tour

Puisque c'est la jonction 1_5 qui pose problème, il suffit ici de modifier la causalité intégrale de J_2 en causalité dérivée pour obtenir le bond graph causal de la Figure 79.

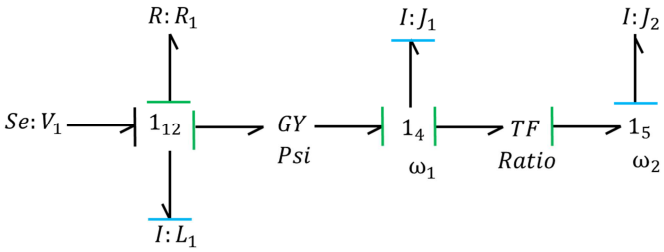


Figure 79. Bond graph causal obtenu au second tour

Notre méthode s'arrête ici, après avoir produit un bond graph causal. La causalité n'est pas unique, mais la première situation de causalité vérifiant les règles est acceptée. Notre programme propage la causalité de jonctions en jonctions, en partant de la première ligne de la matrice. Si nous avions démarré de la dernière ligne, qui correspond à la jonction 1_5, c'est la jonction 1_4 qui n'aurait pas respecté la règle. Nous donnons ci-dessous la matrice BG du bond graph que nous obtenons. Les lignes de cette matrice correspondent de haut en bas aux jonctions 1_{12}, 1_4 et 1_5. Les colonnes correspondent de gauche à droite aux jonctions 1_{12}, 1_4 et 1_5 puis aux éléments V_1, R_1, L_1, GY_1, M_1, M_2 et TF_1.

$$BG = \begin{pmatrix} 0 & 0 & 0 & -11 & 11 & 1 & 11 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -11 & 1 & 0 & 11 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 & -1 \end{pmatrix}$$

6.6.3.6 *Vérification*

D'après la forme de la matrice *BG* du motoréducteur, et sans regarder les tableaux associés, nous savons que le bond graph dispose de trois jonctions (trois lignes) et de sept éléments (10 colonnes moins trois qui correspondent aux liaisons). Nous savons aussi qu'aucune liaison n'existe entre les jonctions. Analysons maintenant les lignes une par une.

La première ligne de la matrice *BG* permet de savoir que :

- Un élément Y_0 impose un effort sur la jonction X_0 et que le transfert de puissance s'effectue de Y_0 vers X_0.
- La jonction X_0 impose un flux sur un élément Y_1 et que le transfert de puissance s'effectue de X_0 vers Y_1.
- La jonction X_0 impose un effort sur un élément Y_2 et que le transfert de puissance s'effectue de X_0 vers Y_2.
- La jonction X_0 impose un flux sur un élément Y_3 et que le transfert de puissance s'effectue de X_0 vers Y_3.

Ces informations permettent de construire la structure autour de la jonction X_0. Nous donnons cette structure sur la Figure 80.

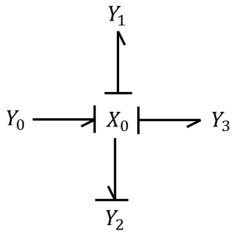


Figure 80. Bond graph déduit de la première ligne de la matrice *BG* du motoréducteur

La seconde ligne de la matrice *BG* permet de savoir que :

- Un élément Y_3 impose un effort sur la jonction X_1 et que le transfert de puissance s'effectue de Y_3 vers X_1.
- La jonction X_1 impose un effort sur un élément Y_4 et que le transfert de puissance s'effectue de X_1 vers Y_4.
- La jonction X_1 impose un flux sur un élément Y_6 et que le transfert de puissance s'effectue de X_0 vers Y_6.

Ces informations permettent de construire la structure autour de la jonction X_1. Nous donnons cette structure sur la Figure 81.

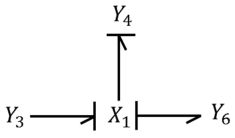


Figure 81. Bond graph déduit de la seconde ligne de la matrice *BG* du motoréducteur

La troisième ligne de la matrice *BG* permet de savoir que :

- Un élément Y_6 impose un effort sur la jonction X_2 et que le transfert de puissance s'effectue de Y_6 vers X_2.
- La jonction X_2 impose un effort sur un élément Y_5 et que le transfert de puissance s'effectue de X_2 vers Y_5.

Ces informations permettent de construire la structure autour de la jonction X_2. Nous donnons cette structure sur la Figure 82.

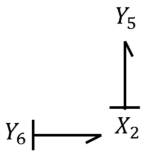


Figure 82. Bond graph déduit de la troisième ligne de la matrice *BG* du motoréducteur

Les trois structures permettent de construire le bond graph global, qui est donné par la Figure 83. Les informations contenues dans le tableau associé aux jonctions et le tableau associé aux éléments permettent de déterminer les types des jonctions et les types et noms des composants.

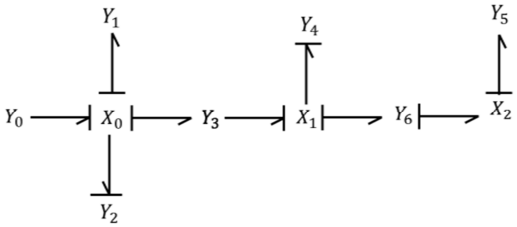


Figure 83. Bond graph global déduit à partir de la matrice *BG* du motoréducteur

6.7 Conclusion du chapitre

Au cours de ce chapitre nous avons présenté une classe permettant de générer automatiquement un modèle équivalent au bond graph d'un macro système électromécanique à partir d'un modèle de bas niveau. Nous avons aussi défini un modèle de bas niveau spécifique, la netlist électromécanique, qui est le point de départ de notre méthode. De futurs travaux pourraient évidemment consister à instancier automatiquement le bond graph correspondant à une netlist électromécanique. Il faudrait pour cela utiliser les primitives de la bibliothèque SystemC AMS SCAX développée par Torsten Mähne [Mähne 11]. Ce travail a néanmoins fait l'objet d'une première publication [Bousquet 13].

7. Cas d'étude : modélisation à haut niveau d'un système éolien

La demande d'énergie à travers le monde est de plus en plus importante. Les sources d'énergies renouvelables sont de plus en plus exploitées. Parmi ces différentes sources, nous pouvons citer l'énergie solaire, l'énergie éolienne, l'énergie hydraulique et l'énergie géothermique. Les énergies éoliennes et hydrauliques ont été utilisées depuis l'Antiquité ; les moulins à vent ou à eau en sont des applications. Ces systèmes exploitant l'énergie cinétique ont été remplacés par les turbines éoliennes ou hydrauliques depuis la découverte des lois du magnétisme.

Mais ce n'est que depuis une vingtaine d'années que l'éolien connait un essor considérable. La production mondiale d'énergie éolienne a été multipliée par soixante sur cet période, passant de moins de cinq gigawatts en 1995 à près de trois cent gigawatts en 2012 [TheWindPower 13]. Au cours de ce qui sera peut-être la « Troisième révolution industrielle », les éoliennes occuperont une place cruciale dans ce que l'on appelle les smart grids, ou réseaux de distribution d'électricité intelligent. Un tel réseau utilise les moyens informatiques afin d'optimiser son efficacité énergétique.

Certes, une éolienne ne pas peut véritablement être considérée comme un système embarqué. Toutefois, elle intègre des parties de contrôle et de commande qui interagissent avec des parties électriques et mécaniques, elles-mêmes déjà liées par l'alternateur ou le générateur électrique. Une éolienne est donc un système hétérogène aux sens multiphysique du terme. Elle est aussi analogique-numérique. De plus, le prototypage virtuel prend ici toute son importance : les ingénieurs ne peuvent pas se permettre d'attendre qu'une éolienne soit complètement assemblée sur site pour développer et tester la partie électronique embarquée de contrôle et de commande. Des outils de prototypage virtuel multiphysique apparaissent nécessaires.

Ce chapitre se décompose en trois parties. La première partie est une introduction aux systèmes éoliens. La seconde partie présente un modèle de haut niveau d'une éolienne. La troisième partie montre des résultats de simulation.

7.1 Les systèmes éoliens

Une éolienne permet de transformer l'énergie cinétique du vent en énergie mécanique puis, le plus souvent, en énergie électrique. Il arrive que l'énergie mécanique soit utilisée pour pomper de l'eau. Les systèmes permettant de produire de l'électricité sont appelés aérogénérateurs. Dans ce qui suit, nous nous intéressons spécifiquement à cette catégorie de systèmes éoliens. De même, bien qu'il existe des éoliennes à axe vertical, nous nous focaliserons sur les éoliennes à axe horizontal, qui sont les plus répandues.

Une éolienne à axe horizontal est constituée d'un mât, qui permet de placer le rotor lent en hauteur, où le vent est plus fort et plus régulier qu'au sol. Au sommet du mât, une nacelle est installée. Elle permet d'abriter le générateur que l'on appelle ici aérogénérateur et la partie de

contrôle et de commande. Elle peut souvent effectuer un mouvement de rotation autour de l'axe du mât enfin de permettre aux pales d'être orientées face au vent. Le rotor lent est constitué de plusieurs pales. Il entraine directement ou par l'intermédiaire d'un système d'engrenage et d'un rotor rapide une génératrice.

7.1.1 Limite de Betz

La puissance du vent contenue dans un cylindre de section *S* est définie par l'équation (76), avec ρ : masse volumique de l'air et v : vitesse du vent.

$$P_{cinétique} = \frac{\rho . S . v^3}{2} \tag{76}$$

Il est d'abord évident que plus une éolienne freine le vent, plus elle récupère d'énergie. Cependant, le ralentissement du vent fait que l'éolienne récupère moins d'énergie. D'après les travaux d'Albert Betz la vitesse du vent est divisée par trois entre l'amont et l'aval des pales, alors que la puissance théorique maximale récupérable par l'éolienne est égale à $\frac{16}{27}$ de la puissance du vent qui « traverse » l'éolienne. Ainsi, nous obtenons l'équation (77), qui donne la puissance maximale théorique récupérable par une éolienne.

$$P_{MAX} = \frac{16 . \rho . S . v^3}{54} \tag{77}$$

7.1.2 Eolienne à vitesse fixe

Les éoliennes à vitesse fixe sont celles dont la conception est la plus simple. La Figure 84 présente une éolienne à vitesse fixe. Les pales sont orientées de façon à ce qu'elles tournent à une vitesse constante. Un multiplicateur permet d'élever la vitesse de rotation et l'arbre rapide entraine une machine asynchrone. On associe souvent une batterie de condensateur afin de relever le facteur de puissance ($\cos \Phi$) du système.

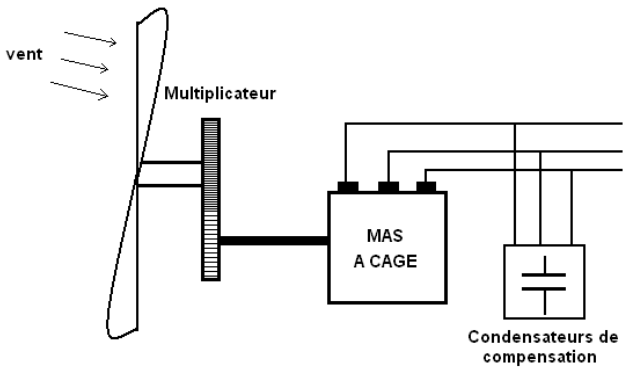


Figure 84. Eolienne à vitesse fixe basée sur un MAS à cage

Cependant, le principal inconvénient de ce type d'éolienne est qu'il ne permet d'atteindre le maximum de la puissance mécanique théorique que pour une seule vitesse de rotation de

l'arbre [Gergaud 02]. De plus, dans un tel système, pour maintenir une vitesse de rotation de l'arbre constante, il faut une très grande réactivité du système de contrôle des pales. En cas de rafales de vent, la puissance électrique varie fortement puisque le système n'est pas capable de s'adapter suffisamment rapidement. La recherche de la vitesse de rotation optimale s'appelle le Maximum Power Point Tracking (MMPT).

7.1.3 Eolienne à vitesse variable

Les éoliennes à vitesse variable permettent un rendement optimal : le rotor rapide tourne à la vitesse permettant d'obtenir la puissance mécanique maximale. Ces systèmes font souvent intervenir des convertisseurs statiques et leurs circuits de commande. Trois principaux types de machines électriques peuvent être utilisés : la machine asynchrone à cage, la machine synchrone à aimants permanents et la machine asynchrone à double alimentation. Dans tous les cas, il est possible d'intercaler des convertisseurs statiques AC/DC puis DC/AC entre la machine et le réseau. Ces systèmes sont plus efficaces que les éoliennes à vitesse fixe, la vitesse de rotation de l'arbre peut varier sur une plage plus importante puisque le convertisseur DC/AC permet de mettre en forme le signal afin de se conformer au réseau. Il y a toutefois un inconvénient important à cette solution : dans le cas de la machine asynchrone à cage (MAS) et de la machine synchrone à aimants permanents (MSAP), l'électronique de puissance doit être dimensionnée afin de pouvoir faire transiter la totalité de l'énergie électrique. Cela provoque des pertes et le système doit être refroidi. Dans le cas d'une machine asynchrone à double alimentation (MADA), le stator est directement relié au réseau, les convertisseurs ne sont dimensionnés que pour faire transiter l'énergie électrique rotorique.

7.2 Un système autonome de type petit éolien

Le petit éolien désigne les systèmes dont la puissance nominale ne dépasse pas 30 KW (100 KW aux Etats-Unis). Ces éoliennes peuvent être raccordées au réseau ou peuvent être utilisées pour stocker de l'énergie dans des accumulateurs. Le système dont nous nous proposons d'étudier la modélisation est une éolienne basée sur une machine synchrone à aimants permanents qui permet de charger une batterie. Un transformateur triphasé est placé en aval de la génératrice. Un redresseur triphasé à diodes est placé entre le transformateur et la batterie. Un anémomètre permet de mesurer la vitesse du vent en amont de l'éolienne. Cette information permet, après traitement, de générer un signal de commande permettant ensuite d'influer sur l'inclinaison des pales. La Figure 85 présente un tel système.

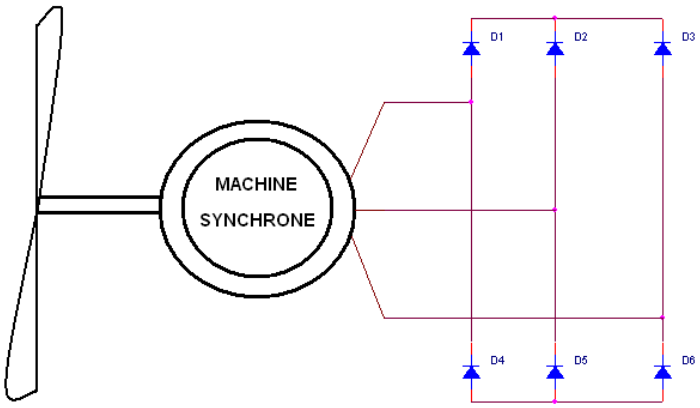


Figure 85. Schéma de principe de l'éolienne

7.2.1 Modèle détaillé du système éolien

Nous allons maintenant entrer dans le détail de chaque partie du système : la partie mécanique, la partie électrique et la partie commande.

7.2.1.1 Partie mécanique de l'éolienne

La partie mécanique permet de générer une vitesse de rotation (et un couple) en fonction de la vitesse du vent. La Figure 86 présente toutes les composantes de la partie mécanique. Ce modèle est tiré de [Bakka 11]. Il aurait été possible d'associer directement un arbre lent sur la génératrice synchrone. T_P correspond au couple exercé par le vent sur une pale. D_P représente l'amortissement dû aux frottements d'une pale, J_P son inertie. Au niveau du moyeu, nous retrouvons l'inertie J_M et l'amortissement D_M. L'association pale-moyeu provoque l'amortissement D_{MP} et la raideur K_{MP}. L'engrenage en lien avec le moyeu a une inertie J_{E1} et cette association provoque l'amortissement D_{ME} et la raideur K_{ME}. Des pertes apparaissent au niveau des engrenages, elles sont notées D_E. L'engrenage en lien avec l'arbre rapide a une inertie J_{E2} et cette association provoque l'amortissement D_{EG} et la raideur K_{EG}. Enfin, la génératrice a une inertie J_G et un amortissement D_G.

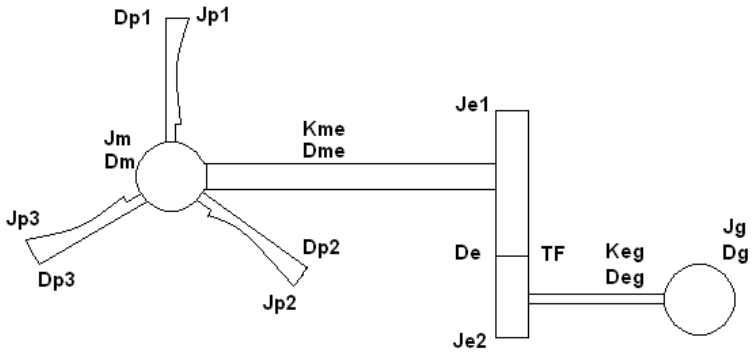


Figure 86. Schéma détaillé de la partie mécanique de l'éolienne

7.2.1.2 Partie électrique de l'éolienne

A partir de la vitesse de rotation de son arbre, la machine synchrone génère un courant triphasé. Nous considérons que la partie de gauche de la Figure 87 correspond au schéma équivalent de la machine synchrone, ce modèle est tiré de [Gergaud 2002]. Celui-ci est constitué (pour une phase) de l'association en série d'une force électromotrice, d'une inductance et d'une résistance. La valeur efficace de la tension E est directement proportionnelle à la vitesse de rotation de l'arbre rapide, comme le montre l'équation (78), où K est le coefficient de Kapp de la machine, N est le nombre de conducteurs d'une phase, p est le nombre de paires de pôles, n est la vitesse de rotation de l'arbre et Φ est le flux maximal pour une spire. C'est cette équation qui permet de faire le lien entre la grandeur mécanique (vitesse de rotation de l'arbre) et la grandeur électrique (force électromotrice générée par la machine synchrone). Un transformateur triphasé est associé à la génératrice puis un pont de diodes triphasé permet de produire un courant continu nécessaire à la charge de batterie. La fréquence est donnée par l'équation (79) où n correspond à la vitesse de rotation de l'arbre et p au nombre de paire de pôles.

$$E = K.N.p.n.\Phi \tag{78}$$

$$f = n.p \tag{79}$$

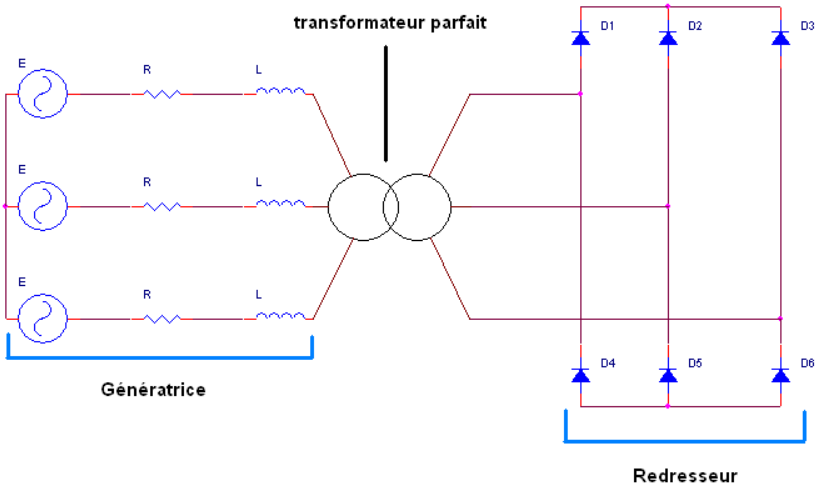


Figure 87. Schéma détaillé de la partie électrique de l'éolienne

Le redresseur double alternance triphasé (non commandé puisqu'il utilise des diodes) fonctionne comme suit : la diode (parmi D_1, D_2 et D_3) dont l'anode est connectée au potentiel le plus élevé conduit, de même que la diode (parmi D_4, D_5 et D_6) dont la cathode est

connectée au potentiel le plus faible. Ainsi, aux tensions de seuil des diodes près, la tension aux bornes de la batterie est égale à la tension entre deux phases. Nous donnons l'ordre dans lequel les diodes conduisent ci-dessous.

Tension charge	u_{12}	u_{13}	u_{23}	u_{21}	u_{31}	u_{32}
Diodes passantes	D_1, D_5	D_1, D_6	D_2, D_6	D_2, D_4	D_3, D_4	D_3, D_5

7.2.1.3 Partie commande de l'éolienne

La partie commande permet de déterminer quelle est la vitesse optimale de rotation de l'arbre en fonction de la vitesse du vent. Ensuite, pour que cette vitesse soit respectée au mieux, la commande agit sur l'angle d'inclinaison des pales. Ainsi, des algorithmes très élaborés de MPPT peuvent être développés. Ce n'est pas l'objectif de notre travail qui se focalise sur le prototypage virtuel des parties matérielles. Nous utiliserons un système de commande rudimentaire.

7.3 Analyse du système en vue de sa modélisation

L'objectif est ici de produire un prototype virtuel de haut niveau du système éolien en se basant sur les concepts présentés tout au long de ce manuscrit. Il est évident que dans un tel système, les informations relatives à la puissance, qui sont cruciales doivent être maintenues au cours de la modélisation. Il faudra donc utiliser des modèles conservatifs ou des modèles plus abstraits qui permettent tout au moins de maintenir l'information de puissance (cf. chapitre 4).

La partie mécanique a pour entrées la vitesse du vent et l'angle d'inclinaison des pales et pour sortie la vitesse de rotation de l'arbre de la machine synchrone. Cette vitesse de rotation est transformée en tension électrique par la génératrice. Nous avons affaire à un système électromécanique. Dans ce manuscrit, nous proposons deux moyens permettant de modéliser les systèmes mécaniques ou électromécaniques : le circuit équivalent électrique et les bond graphs. En particulier, comme nous l'avons remarqué au chapitre 6, le bond graph permet d'abstraire le domaine physique, ce qui peut être vu comme la première étape d'un cheminement bottom-up. C'est pour cela que la modélisation par les bond graphs est ici plus pertinente que la modélisation par circuit électrique équivalent. Toutefois, contrairement au circuit électrique équivalent, nous n'avons pas développé de méthodes permettant de générer automatiquement le modèle mathématique correspondant à un bond graph. Il nous faudra donc utiliser un outil permettant de générer la représentation d'état à partir d'un bond graph (20-Sim). Cependant, nous devons être en mesure de simuler le modèle global de haut niveau (modèle mathématique) au moyen de SystemC AMS. Toutefois, le bond graph de la partie

mécanique pose un problème. En effet, la force qu'exerce le vent sur une pale est modulée par l'angle α qui évolue dans le temps. Nous ne pourrons donc pas abstraire cette partie du bond graph dans la représentation d'état linéaire à coefficients constants qui, seul être traité par un MoC de SystemC AMS. Une idée intéressante consiste à modéliser dans un même bloc la partie commande et la partie mécanique permettant de modéliser la force qu'exerce le vent sur une pale. Cette force sera ensuite la grandeur d'entrée du modèle bond graph de la partie mécanique de l'éolienne.

Nous adoptons donc la modélisation bond graphs pour la majorité de la partie mécanique, voyons ce qu'il en est pour la partie électrique. La partie électrique se divise en deux sous parties ou blocs, l'un est linéaire, l'autre non. Le bloc linéaire est constitué du schéma équivalent de la génératrice et du transformateur. Le bloc non linéaire correspond au redresseur. Le bloc linéaire peut être modélisé au moyen des bond graphs. Cependant, il n'est pas possible de modéliser dans le même bond graph la partie mécanique et la partie électrique linéaire. En effet, la transformation vitesse de rotation – tension électrique sinusoïdale ne peut être modélisée par un transformateur ou un gyrateur. Nous devrons utiliser un bloc TDF intermédiaire permettant d'opérer cette transformation. La partie électrique linéaire peut être modélisée en ELN : elle est suffisamment simple et n'engendre que peu d'équations.

La partie non linéaire pose des problèmes quant à sa modélisation sous SystemC AMS. En effet, les modèles linéaires conservatifs peuvent être traités au moyen du MoC LSF (schéma blocs) ou du MoC ELN (circuit électrique équivalent), mais lorsqu'il s'agit de créer des modèles non linéaires conservatifs avec SystemC AMS, la tâche se complique. Le MoC TDF, le seul qui permette une totale liberté au concepteur n'offre que très peu de primitives dans la bibliothèque. Or, des primitives telles que celles qui sont proposées par la bibliothèque LSF s'avèreraient nécessaires. Le style de modélisation LSF encapsulé pourra s'avérer être ici d'une grande utilité, même si des blocs TDF purs devront y être associés.

7.4 Modèle haut niveau de l'éolienne

L'objet de ces paragraphes est de présenter les modèles de haut niveau des différentes parties du système éolien.

7.4.1 Modèle de la partie « vent – commande – pale »

Comme nous l'avons expliqué plus haut, l'angle α n'étant pas constant dans le temps, cette partie ne peut pas être exprimée sous forme de représentation d'état linéaire. L'entrée de ce bloc est la forme d'onde du vent, c'est à dire l'évolution de la vitesse du vent en fonction du temps, la sortie est la force s'appliquant sur une pale. Un modèle non linéaire à variables d'état peut être construit, mais dans la version actuelle de SystemC AMS, il n'existe pas de MoC adapté au traitement de ce type de modèle.

Théoriquement, en fonction de la vitesse du vent, l'angle d'inclinaison des pales à atteindre pour obtenir la vitesse de rotation optimale doit être calculé. Ainsi, pour disposer d'un

système de commande fiable, trois grandeurs doivent être mesurées : la vitesse du vent, la vitesse de rotation de l'arbre et l'angle d'inclinaison des pales. Nous nous contenterons d'un système simplifié où nous considérons que la vitesse de rotation de l'arbre optimale est connue si l'on connait la vitesse du vent. Ainsi, la mesure du vent à un instant *t* permet de définir la vitesse de rotation optimale de l'arbre à ce même instant *t*. Nous ne pouvons pas envisager que l'étape d'action sur le système, soit le pivotement des pales, va s'opérer de façon instantanée. Nous considérons ici que l'angle théorique est atteint de façon linéaire en fonction du temps : il peut varier d'un degré à chaque seconde, par exemple. Ainsi, l'angle α théorique optimal à l'instant *t* ne sera atteint qu'à un instant *t+x*. Lorsque la vitesse du vent varie plus lentement que la « constante de temps » du système de commande des pales, le comportement de notre modèle est raisonnable. En cas de rafales de vent ou simplement d'une évolution de la vitesse du vent plus rapide que la constante de temps des pales, notre modèle n'est plus performant.

Notre modèle est donc constitué d'un bloc LSF de type correcteur proportionnel intégral permettant de générer la force s'exerçant sur une pale en fonction de la vitesse du vent. L'entrée de ce bloc est la vitesse du vent, la sortie est le couple effectif appliqué par le vent sur une pale

7.4.2 Modèle de la partie mécanique « pales – alternateur»

Cette partie est constituée de la mécanique du système. Nous considérons que le couple exercé par le vent sur une pale est proche du couple permettant le rendement maximum. La Figure 88 montre le bond graph de la partie concernée. Ce bond graph a ensuite été abstrait afin d'obtenir la représentation d'état qui a été utilisée comme modèle de haut niveau. Ce bloc comporte une entrée (le couple appliquée sur une pale) et une sortie : la vitesse de rotation de l'arbre rapide.

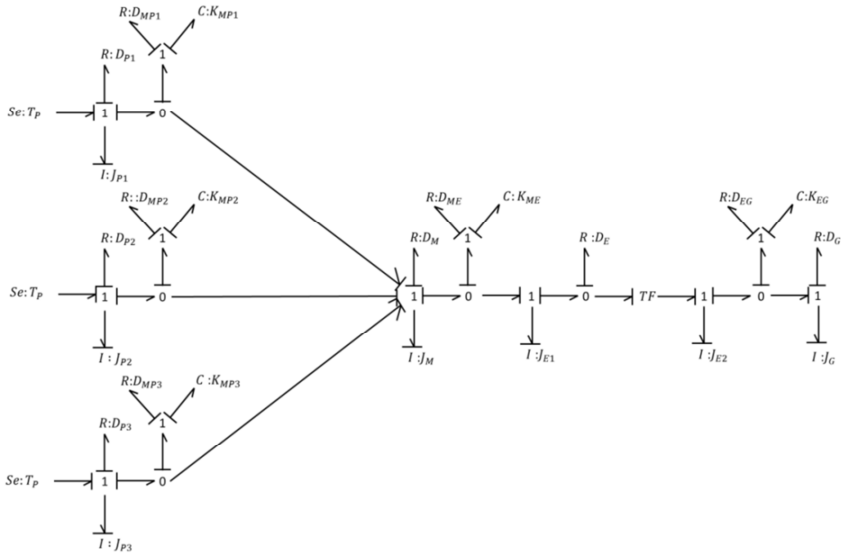


Figure 88. Bond graph de la partie linéaire du système

7.4.2 Modèle de la partie électrique linéaire

Cette partie est constituée du modèle équivalent de l'induit de la machine synchrone ainsi que du transformateur qui y est associé. Nous avons décidé d'utiliser le MoC ELN pour la modéliser. En effet : ce bloc doit disposer des deux entrées que sont l'amplitude et la fréquence du signal électrique produit. Il doit aussi disposer de six sorties : les trois tensions phase-neutre (ou tensions simples) ainsi que les courants de phase. C'est pourquoi le MoC ELN est ici une solution intéressante : trois terminaux électriques de sortie suffiront là où il faudrait utiliser six signaux LSF pour transmettre les mêmes informations. Ceci est une astuce afin d'alléger le travail de description en SystemC AMS.

7.4.3 Modèle du redresseur

La partie linéaire vue précédemment dispose de trois terminaux de sortie : les trois phases du secondaire du transformateur. Nous avons donc trois couples courant-tension. La sortie du redresseur est la tension monophasée qui peut être appliquée à la charge. Le code du redresseur est donné en annexe 3. Le module reçoit en entrée les trois tensions simples V_1, V_2 et V_3 (sous forme de signaux TDF, une conversion ELN/TDF étant réalisée préalablement). Il identifie ensuite la tension la plus importante et la tension la plus faible pour en donner la différence et l'écrire sur un port TDF de sortie.

7.4.4 Modèle global

La Figure 89 présente le modèle global du système. L'entrée du système est bien entendu la vitesse du vent. T_P correspond au couple s'exerçant sur une pale. C'est le couple effectif, nous considérons que l'angle optimal des pales est calculé à partir de la vitesse du vent. *n* correspond à la vitesse de rotation de l'arbre rapide, celui qui entraine la génératrice. Il est obtenu à partir de T_P par un bloc de type représentation de transfert. Les grandeurs *A* et *f* correspondent à l'amplitude et à la fréquence de la force électromotrice de la génératrice synchrone. Elles sont obtenues à partir de *n*. V_1, V_2 et V_3 sont les tensions simples fournies par le modèle de la machine synchrone. Enfin, ces tensions sont redressées afin d'obtenir *uout*, tension monophasée de sortie. Les différentes couleurs font apparaître les MoCs utilisés. Le bleu correspond au TDF, le rouge au LSF et le vert à ELN. Les carrés noirs en entrée d'un bloc représentent un convertisseur. Il n'est pas nécessaire de placer des convertisseurs en entrée du bloc ELN modélisant la machine synchrone : *A* et *f* sont des paramètres d'entrée de ce bloc ELN. Comme annoncé au chapitre 3, nous voyons que niveaux d'abstraction et modèles de calcul ne sont pas liés puisque nous avons réussi à modéliser à haut niveau un système en utilisant les trois MoCs.

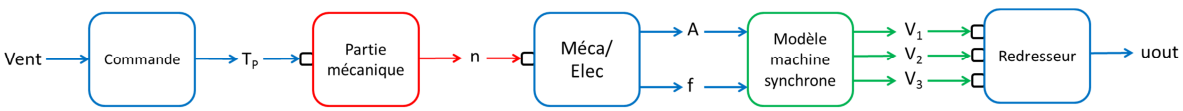


Figure 89. Diagramme du modèle de l'éolienne

7.5 Simulations

La Figure 90 présente des résultats de simulations. Le profil de la vitesse du vent choisi évolue lentement : nous n'avons pas fait apparaître de variations rapides que notre « commande » n'aurait pas été capable de traiter. La force (ou couple) s'exerçant sur une pale est présentée sur le second graphe. La force optimale apparaît en rouge, la force effective apparaît en vert. Nous modèle permet de modéliser l'inertie du système par rapport à la commande : l'angle optimal des pales n'est obtenu instantanément. Le troisième graphe présente les tensions entre les phases et le neutre (tensions simples). Enfin, le quatrième graphe présente la tension de sortie du redresseur. Ces deux derniers graphes sont des agrandissements.

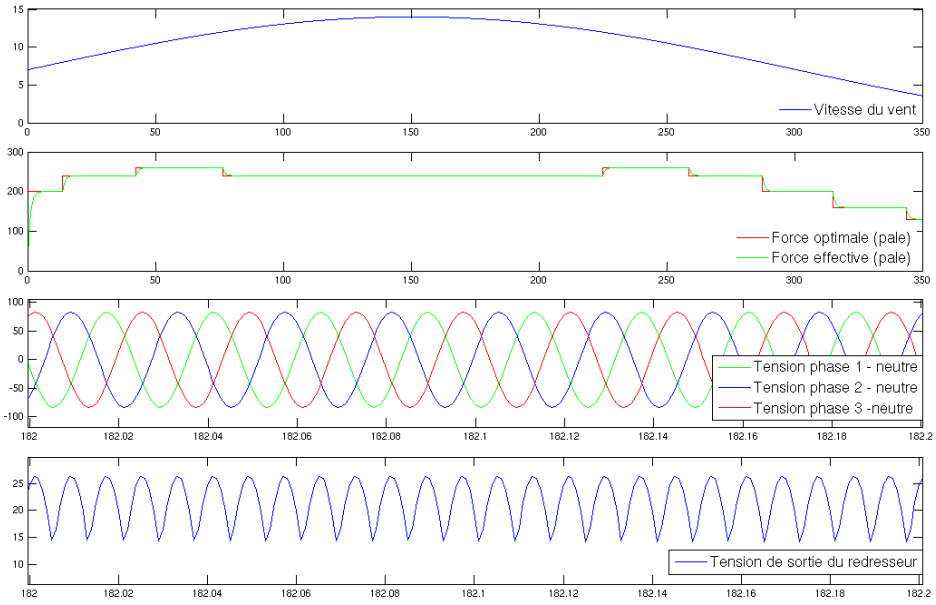


Figure 90. Simulation de l'éolienne - Comportement

Il est aussi possible d'enrichir le modèle global de la Figure 89 avec des informations permettant d'obtenir la puissance utile en sortie ou les pertes à différents niveaux du système. La Figure 91 montre les résultats obtenus à partir du même profil de vent. Les second et troisième (agrandissement) graphes montrent la puissance utile en sortie. Nous avons placé une charge de type résistive à la sortie du redresseur double alternance, sans ajouter de condensateur. Pour cela, nous avons utilisé le signal TDF *uout* pour créer une source de tension ELN qui débite sur la charge. Les pertes par frottements au niveau de l'engrenage sont obtenues différemment. Une représentation d'état nous permet de modéliser la partie mécanique, nous avons donc enrichi cette représentation d'état avec les grandeurs permettant d'estimer les pertes, comme expliqué au chapitre 4. Un bloc TDF doit ensuite être utilisé pour effectuer le calcul

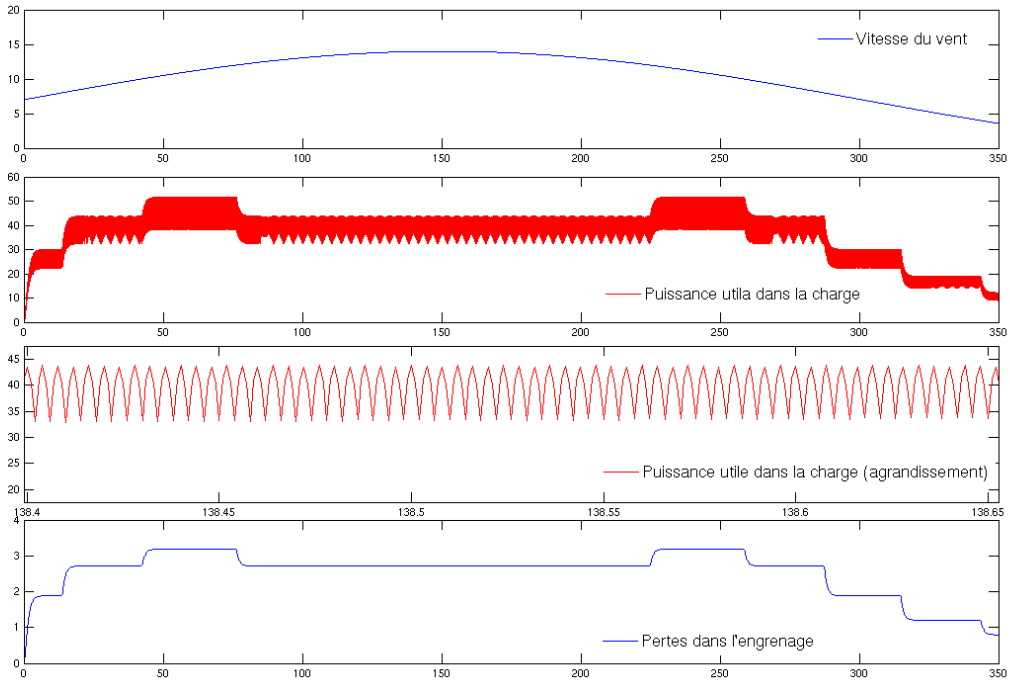


Figure 91. Simulation de l'éolienne – Aspects puissance

7.6 Conclusion du chapitre

Au cours de ce chapitre, nous avons traité un exemple d'application multiphysique. Il a été nécessaire de s'appuyer sur différents outils (SystemC AMS et 20-Sim). 20-Sim nous a permis de générer la représentation d'état de la partie mécanique à partir du bond graph. Puis cette représentation d'état a été intégrée dans le modèle complet SystemC AMS.

Au chapitre 3, la modélisation du four avait été réalisée de quatre façons différentes : entièrement en TDF, en LSF, en LSF encapsulé ou en ELN. Bien qu'ayant allégé la modélisation de l'éolienne en simplifiant de façon considérable la partie commande, ce système demeure bien plus complexe que le four. Il apparaît que la possibilité de combiner au sein d'un même modèle les différents MoCs est un atout considérable. Le modèle de la machine synchrone triphasée a été très simple à implémenter grâce à cela.

8. Perspectives

Un travail de recherche ne s'achève jamais véritablement. De nouvelles pistes sont toujours à explorer. D'autre part, les conférences permettent des discussions avec des gens travaillant dans des domaines divers où la modélisation à haut niveau peut s'avérer utile. Nous énumérons ici certains axes d'approfondissement de ce travail de thèse mais aussi de nouveaux travaux envisageables.

8.1 Enrichir les modèles de haut niveau avec d'autres informations

Dans ce document, nous avons illustré comment enrichir des modèles de haut niveau avec la consommation. Il peut aussi s'avérer pertinent d'intégrer une information telle que la surface sur silicium occupée par un circuit. Si l'on admet que la surface utilisée par une capacité élémentaire de 10 nF est de *x*, alors on est capable de déterminer la surface totale occupée par les capacités d'un circuit. La même démarche peut être appliquée aux autres composants. La surface occupée par le circuit, information de bas niveau par excellence peut alors être ramenée à haut niveau et l'on peut compléter la représentation d'état avec une donnée de type `double`.

8.2 Affiner les modèles

La bibliothèque « officielle » SystemC AMS propose seulement le nullor (`sca_eln::sca_nullor`) pour modéliser l'amplificateur opérationnel. Les limites de cette primitive sont rapidement atteintes. Ce modèle ne permet que la modélisation de l'amplificateur opérationnel en régime linéaire, de plus, celui-ci est considéré comme étant parfait. Dans sa thèse, Franck Paugnat a développé un modèle qui se rapproche d'avantage de celui d'un amplificateur opérationnel réel. Ce modèle fait apparaître le produit gain-bande, le slew rate, la saturation... D'autre part, il est possible d'utiliser ce modèle pour des circuits comparateurs. Il serait intéressant d'utiliser les équations définissant ce modèle dans notre méthode d'abstraction. Tous les facteurs linéaires (limite en fréquence, slew rate) peuvent être introduits sans problème. Nous pouvons aussi songer à affiner les modèles des composants passifs en faisant intervenir les effets de la température. Là encore, il faudra savoir « se brider », l'intérêt des modèles de haut niveau étant de permettre des simulations rapides, il ne faut pas surcharger de détails les modèles. Une analyse des spécifications bien faite permet d'identifier clairement les paramètres critiques et donc le niveau de détail minimum. Ensuite, une bonne connaissance du langage et du simulateur permet d'établir le niveau de détail maximum au-dessus duquel les durées de simulations ne sont plus raisonnables.

8.3 Du bond graph à la représentation d'état

Nous avons développé une méthode permettant de passer d'un modèle de bas niveau de type netlist électromécanique au bond graph, l'étape suivante, du bond graph à la représentation d'état existe déjà dans différents outils spécifiques. Cependant, afin d'éviter d'avoir à utiliser conjointement plusieurs outils, il pourrait s'avérer utile de disposer d'une telle méthode implémentée sous SystemC AMS.

8.4 Etendre les travaux à d'autres domaines physiques

Nos outils permettent de travailler efficacement dans les domaines électriques et mécaniques. Les systèmes hétérogènes évoluent dans tous les domaines physiques. Nous pouvons envisager d'étendre nos méthodes à l'hydraulique ou à la chimie par exemple. Le cas de l'hydraulique est très simple à résoudre car les analogies avec les systèmes électriques peuvent être utilisées. Un débit hydraulique en m^3/s représente pour l'eau exactement la même chose qu'un courant électrique en Ampères ou Coulombs/s pour l'électricité. Par contre, le domaine de la chimie s'avérera certainement plus ardu à modéliser.

8.5 Etablir/améliorer le lien avec les langages de spécifications

Des langages de très haut niveau comme UML (Unified Modeling Language) ou SysML (Systems Modeling Language) permettent la spécification, l'analyse, la conception, la vérification et la validation des systèmes. A l'issue de la phase de spécification, ils peuvent même générer les prototypes C++ des fonctions définies. Des travaux ont été entrepris [Ferrari 12] afin de coupler SysML à SystemC AMS, ceci menant à DesyreML. Un tel outil peut s'avérer précieux pour le traitement des systèmes complexes et les systèmes de systèmes.

8.6 Modélisation des comportements non linéaires

Enfin, différents axes d'améliorations sont possibles au niveau de SystemC AMS. Le premier concerne évidemment les composants non linéaires tels que la diode, le transistor et les autres semi-conducteurs. Cela permettrait de disposer de la presque totalité du flot de conception électronique sous SystemC/SystemC AMS. Il ne manquerait plus que le niveau silicium. Des résultats ont déjà été publiés à propos d'un MoC et d'un solveur non linéaires. Il serait crucial d'intégrer ces fonctionnalités dans une version prochaine. Dans l'état actuel des choses, les limites de SystemC AMS sont rapidement atteintes. En effet, les systèmes, pour être résolus, doivent être décrits à partir d'équation différentielles linéaires ou sous forme algorithmiques. Une astuce utilisée par Franck Paugnat pour son modèle ELN d'amplificateur opérationnel simplifié a été de partitionner son modèle en fonction linéaires. On se retrouve donc avec une linéarité par morceaux. Cependant, la matrice d'équations d'un circuit est résolue en une fois, par une suite de calculs. Les valeurs des variables devant être tracées ne sont écrites dans le fichier de trace qu'une fois que la matrice d'équations a été entièrement résolue. Il est donc possible d'utiliser des fonctions de rappel afin de corriger ces valeurs avant de les écrire dans le fichier. Il faudrait pouvoir être capable d'interrompre le calcul de la matrice d'équations et d'exécuter une fonction de rappel afin de corriger éventuellement la valeur de la variable résolue. L'impossibilité actuelle de modéliser les non linéarités de manière simple constitue un handicap rédhibitoire pour SystemC AMS par rapport aux autres langages.

8.7 Réutilisation du produit de l'élaboration

Ce point concerne aussi le langage SystemC AMS. Dans les simulations de modèles ELN ou LSF, nous avons vu au chapitre 3 que la phase d'élaboration représente une trop grande partie du temps de simulation total. Ici, il faudrait être capable de séparer les deux phases,

élaboration et simulation. Dans le cadre d'un système hétérogène analogique-numérique, entre deux simulation, si l'utilisateur n'apporte aucune modification à la partie analogique (surtout si elle est modélisée en ELN ou LSF), l'élaboration ne devrait pas être reproduite. Ceci permettrait de gagner beaucoup de temps en termes de durées de simulation.

8.8 Modélisation des phénomènes analogiques dans les circuits numériques

Un circuit numérique est constitué de portes logiques et donc de transistors. C'est un circuit analogique non linéaire. SystemC AMS peut donc être utilisé afin de modéliser les phénomènes analogiques se produisant à l'intérieur des blocs numériques. Franck Paugnat a largement traité la question en ce qui concerne les temps de propagation et les transitions notamment. Nous aborderons donc ici brièvement les problèmes liés à la sûreté de fonctionnement. Imaginons que l'on souhaite modéliser les effets d'un glitch (défaillance ou interruption dans le signal) au niveau de l'alimentation V_{DD} sur un circuit numérique. Ceux-ci ne sont pas de nature numérique, mais plutôt analogique. Nous obtiendrons des phases transitoires ayant la forme d'un premier ordre ou d'un second ordre. La Figure 92 fait apparaître en rouge ces différentes phases transitoires. Il est possible de modéliser ces phénomènes avec VHDL-AMS mais le couple SystemC/SystemC AMS TDF devrait s'avérer plus efficace. En effet, la possibilité d'utiliser des fonctions de transfert prédéfinies simplifiera considérablement le problème.

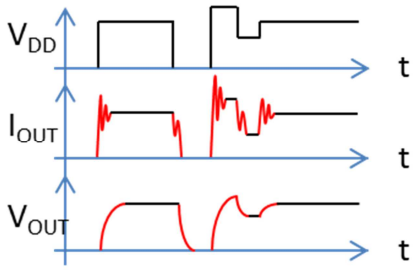


Figure 92. Exemple de comportements analogiques dans les systèmes numériques

9. Conclusions

La première partie du travail réalisé au cours de cette thèse a porté sur une analyse des possibilités offertes par SystemC AMS. Nous y avons défini quel style de modélisation était le plus pertinent en fonction de la taille du modèle et de l'ampleur de la simulation. Il ressort de ce travail que pour un système donné, le développeur SystemC AMS doit être capable d'évaluer le nombre d'équations engendrées par le modèle qu'il souhaite créer. Ceci n'est possible que s'il a une bonne connaissance des modèles de calculs sur lesquels reposent les styles de modélisation.

Ensuite, nous avons introduit une méthode permettant de générer automatiquement la représentation d'état d'un circuit électrique linéaire à partir de sa netlist. Dans le cadre de la simple validation à haut niveau du comportement d'un circuit, seule la (ou les) sorties du circuit sont considérées comme des informations fonctionnelles et donc incorporées au modèle de haut niveau. Ainsi, la simulation de la représentation d'état est plus rapide que celle du circuit, puisque le modèle contient moins d'informations : nous obtenons un modèle non conservatif. Nous avons ensuite enrichi le modèle de haut niveau avec des informations permettant une évaluation de la consommation de puissance. Ceci s'inclut dans une approche d'aide à la conception descendante (raffinement d'une fonction de transfert, exploration architecturale) puis ascendante (abstraction, extraction d'informations pertinentes) qui permet, en partant d'une fonction de transfert d'obtenir un circuit correspondant puis, à partir de ce circuit, de générer une représentation d'état incluant les informations permettant le calcul de la consommation. Cela est réalisé en incorporant au vecteur de sortie de la représentation d'état certains courants et potentiels du circuit. Bien que le modèle obtenu permette d'évaluer la consommation de puissance du circuit, il demeure non conservatif. Produire des modèles de haut niveau conservatifs au moyen de cette méthode n'est pas pertinent puisque cela consisterait à incorporer tous les courants et potentiels du circuit dans le vecteur de sortie de la représentation d'état. Nous aurions exprimé sous forme mathématique le fonctionnement d'un circuit, mais le modèle conservatif ainsi obtenu n'en serait pas pour autant un modèle de haut niveau. De plus, la durée de simulation de ce modèle serait équivalente à celle du circuit. C'est alors qu'apparaît un concept crucial en modélisation : le développeur ou l'ingénieur doit être capable d'établir deux bornes définissant le niveau de détails du modèle qu'il souhaite obtenir. Ces bornes sont définies par les spécifications du système qui doit être vérifié d'une part et par les spécificités du langage qu'il utilise, d'autre part.

Nous avons ensuite étendu l'approche précédente aux systèmes électromécaniques. Les spécialistes de la physique travaillent à très bas niveau sur des outils d'analyse par éléments finis. Ils produisent des modèles analytiques qui peuvent être traduits en circuits électriques équivalents. Par l'intermédiaire du circuit électrique équivalent, nous avons pu générer des modèles de haut niveau de systèmes électromécanique. Le modèle mathématique prend ici encore la forme d'une représentation d'état. Ceci permet d'envisager des simulations globales de systèmes hétérogènes et multiphysiques à haut niveau en utilisant SystemC AMS. Là aussi,

nous avons pu intégrer dans le modèle de haut niveau des informations permettant la modélisation des échanges d'énergie.

Puis, dans un souci de généraliser ces approches pour les systèmes linéaires, nous avons utilisé les bond graphs. Une représentation de bas niveau similaire à une netlist a été définie afin de pouvoir modéliser les systèmes électromécaniques. A partir de cette représentation, un modèle équivalent à un bond graph est automatiquement créé. Ce travail pourrait se poursuivre par l'instanciation automatique d'un bond graph sous SystemC AMS au moyen de la bibliothèque SCAX [Mähne 11].

Enfin, une étude de cas a été présentée : un système petit éolien. Les différents concepts de la modélisation y ont été illustrés. En l'état actuel des choses, SystemC AMS ne permet pas de réaliser facilement des modèles conservatifs de systèmes non linéaires à bas niveau puisque les composants de base que sont la diode et les transistors ne sont pas intégrés dans la bibliothèque. Pour modéliser un sous-système tel que le redresseur, il a fallu composer avec deux blocs de haut niveau, l'un permettant de calculer la tension de sortie et l'autre le courant.

Le besoin de simuler dans leur globalité les systèmes hétérogènes et multiphysiques afin de valider leur fonctionnement nécessite soit la cosimulation du système au moyen de plusieurs outils soit une simulation globale ne faisant intervenir qu'un outil unique permettant l'usage de différents styles de modélisation basés sur des modèles de calculs divers. Or, il apparaît clairement que les systèmes embarqués devenant de plus en plus complexes, les cosimulations seront de plus en plus délicates à mettre en œuvre. C'est donc la deuxième option qui sera à mon avis retenue : elle permet en effet de s'affranchir des problèmes de communications entre les différents outils. Cependant, pouvoir simuler une puce dans son intégralité avec un niveau de détail très élevé, en un temps raisonnable et au moyen d'un outil unique restera un rêve pour les ingénieurs, du moins pour un certain nombre d'années encore…

Une possibilité intéressante est de partitionner le système afin de développer séparément chacune de ces parties jusqu'à un très grand niveau de détail – approche classique top-down, la méthode actuelle – puis d'abstraire les modèles de bas niveau obtenus afin de pouvoir envisager une simulation globale n'impliquant qu'un seul outil. Les méthodes d'abstractions sont spécifiques à chaque domaine mais pour un domaine donné, la méthode d'abstraction peut être générique. Dans le cas des systèmes linéaires, nous avons vu que le modèle abstrait peut se présenter sous la forme d'une représentation d'état, un modèle bond graph servant d'intermédiaire entre le modèle de bas niveau et le modèle mathématique. La représentation d'état ainsi obtenue peut être utilisée sous SystemC AMS afin de pouvoir simuler les parties linéaires (analogiques ou physiques) conjointement avec la partie numérique, matérielle et/ou logicielle. De plus, ce type de modélisation permet d'intégrer des informations qui ne sont pas habituelles, bien qu'intéressantes à haut niveau. Ainsi, dans ce mémoire nous avons illustré comment enrichir la représentation d'état avec les informations permettant d'évaluer la consommation.

Enfin, bien que des entreprises des secteurs de l'électronique et de la microélectronique aient contribué au développement de SystemC AMS, il semble qu'il soit trop tôt pour dire si ce langage s'étendra progressivement dans l'industrie. Du côté académique, en France, il n'est à ma connaissance enseigné qu'à la seule Université Pierre et Marie Curie et il est introduit à Télécom Physique Strasbourg. Toutefois, si ce langage ne s'impose pas, les concepts qu'il met en œuvre devraient être repris dans le futur.

Les travaux présentés aux chapitres 3, 4 et 6 de ce mémoire ont été publiés et les travaux présentés aux chapitres 5 et 7 seront l'objet de publications futures.

.

Annexes

Annexe 1

Nous présentons ici certains exemples de code relatifs au chapitre 4.

Annexe 1.1

Ce code présente une cellule de Sallen-Key en SystemC AMS.

```
SC_MODULE(sallen_key_lowpass)
{
      sca_elec_port in, out;
      sc_in <double> r1, r2, c1, c2;
      sca_elec_ref gnd;
      sca_elec_node mid, pi;
      sca_sc2r *r_1, *r_2;
      sca_sc2c *c_1, *c_2;
      sca_nullor *null;

      SC_CTOR(sallen_key_lowpass)
      {
            r_1 = new sca_sc2r("r_1");
            r_1->p(in);
            r_1->n(mid);
            r_1->ctrl(r1);

            r_2 = new sca_sc2r("r_2");
            r_2->p(mid);
            r_2->n(pi);
            r_2->ctrl(r2);

            c_2 = new sca_sc2c("c_2");
            c_2->p(pi);
            c_2->n(gnd);
            c_2->ctrl(c2);

            c_1 = new sca_sc2c("c_1");
            c_1->p(mid);
            c_1->n(out);
            c_1->ctrl(c1);

            null = new sca_nullor("null");
            null->nip(pi);
            null->nin(out);
            null->nop(out);
            null->non(gnd);
      }
};
```

Annexe 1.2

Ce permet le calcul des composants de chacune des cellules de Sallen-Key passe-bas.

```
void comput_cells()
{
      if (type == "lowpass")
      {
            int i, j;
            j=1;
            i=0;
            while(j <=ncells)
            {
                  r1(i)=1e3;
                  r2(i)=r1(i);
                  c2(i)=1/(2*r1(i)*qp(i)*wp(i));
                  c1(i)=4*qp(i)*qp(i)*c2(i);

                  j +=1;
                  i +=1;
            }
      }
}
```

Annexe 1.3

Ce code présente comment sont instanciées et connectées les cellules de Sallen-Key.

```
void instanciation_lowpass_imp()
{
      for(int i=0; i<ncells; i++)
      {
            low_sallen[i] = new
            sallen_key_lowpass(name_cell[i].c_str(),r1(i),r2(i),c1(i)
            ,c2(i));
      }
      lp = new order1_lowpass("lp_order_1", r0, c0);
      lp->in(node[in]);
      lp->out(in2);
      low_sallen[0]->in(in2);
      low_sallen[ncells-1]->out(node[ncells-1]);
      for(int i=0; i<ncells-1; i++)
      {
            low_sallen[i]->out(node[i]);
      }
      for(int i=1; i<ncells; i++)
      {
            low_sallen[i]->in(node[i-1]);
      }
}
```

Annexe 1.4

Ce code permet d'ajouter une inductance dans le système d'équations préalablement créé.

```
if(name_eq[i].find("stateL")==0 //ecriture equation inductance
{
      for(int k=0;k<n_comp;k++)//balayage de la liste des composants
      {
            if(Comp[k].eq_comp==name_eq[i])//recherche inductance
            {
                  if(name_var[j]=="u"+to_string(Comp[k].p))//recherche
            potentiel interface positive
                  {
                        sys(j,i)=1;
                  }
                  else if(name_var[j]=="u"+to_string(Comp[k].n))
            //recherche potentiel interface negative
                  {
                        sys(j,i)=-1;
                  }
                  if (name_var[j]=="di"+to_string(Comp[k].name))
            //recherche derivee courant inductance
                  {
                        sys(j,i)=-Comp[k].val;
                  }
            }
      }
}
```

Annexe 1.5

Ce code permet d'ajouter l'équation correspondant à la loi des noeud au système d'équations.

```
else if(name_eq[i].find("node")==0)//ecriture KCL
{
      size_t taille = name_eq[i].size() + 1;//recuperation numero
noeud
      char *buffer = new char[taille];
      strncpy( buffer, name_eq[i].c_str(), taille );
      char *nods = new char[taille-4];
      for(int m=4; m<=taille; m++)
      {
            nods[m-4] = buffer[m];
      }
      int nodi;
      nodi=atoi(nods);//ecriture numero nœud type entier
      for (int m=0; m<n_comp; m++)
      {
            if (nodi == Comp[m].p)//recherche composants connectes au
      nœud par interface positive
            {
```

```
                if (name_var[j] == "i"+to_string(Comp[m].name))
                {
                     sys(j,i)=1;
                }
           }
           else if (nodi == Comp[m].n) //recherche composants
     connectes au nœud par interface negative
           {
                if (name_var[j] == "i"+to_string(Comp[m].name))
                {
                     sys(j,i)=-1;
                }
           }
     }
}
```

Annexe 1.6

Ce correspond à l'opération qui consiste à reorganiser les lignes du système d'équations.

```
void shift_lines()
{
     int sum[n_eq_ind];//tableau nombre variables inconnues ligne
     int min;//indice ligne somme variables inconnues minimum
     int k;
     k=0;
     while (k != n_eq_ind)
     {
          trouv_sum(sum);//calcule nombre variables inconnues ligne
          min=get_min(sum,n_eq_ind-k);//recherche ligne inconnues
     minimum
          shift_row(min,k,sum);//permute ligne indice k avec ligne
     indice n
          retrancher_ligne(n_eq_ind-1-k);//n -= 1
          k += 1;
     }
}
```

Annexe 1.7

Ce code montre l'instanciation de la classe permettant une conversion LSF vers TDF.

```
sca_lsf::sca_tdf_sink* lsf2tdf; // déclaration
lsf2tdf = new sca_lsf::sca_sink("lsf2tdf1"); // instanciation
lsf2tdf->x(out_lsf); // connexions
lsf2tdf->y(out);
```

Annexe 1.8

Ce code montre comment multiplier deux signaux TDF.

```
SCA_TDF_MODULE(Multiplicator)
{
      sca_tdf::sca_out <double> out;
      sca_tdf::sca_in <double> in1, in2;
      void processing()
      {
            out.write(in1.read()*in2.read());
      }
      SCA_CTOR(Multiplicator){}
};
```

Annexe 1.9

Ce code montre comment est réalisé le filtre à variable d'état en SystemC AMS

```
SC_MODULE(State_Variable_Filter)
{
      sca_eln::sca_terminal in, out; //declaration des ports
      sca_eln::sca_node_ref gnd; //reference elctrique
      sca_eln::sca_node n2, n3, n4, n5, n6, n7, n8; //noeuds
      sca_eln::sca_r    *r_1,*r_2,*r_3,*r_4,*r_5,*r_6,*r_7,*r_L,*r_o;
//resistances
      sca_eln::sca_c *c_1, *c_2; //capacites
      sca_eln::sca_nullor *null_1, *null_2, *null_3; //nullors
      double r, c; //valeurs des resistances et capacites

      State_Variable_Filter(sc_core::sc_module_name  _n,  double  r_v,
double c_v)
      {
            r = r_v;
            c = c_v;

            r_1 = new sca_eln::sca_r("r_1", r);
            r_1->p(in); r_1->n(n2);

            r_2 = new sca_eln::sca_r("r_2", r);
            r_2->p(n2); r_2->n(n3);

            null_1 = new sca_eln::sca_nullor("null");
            null_1->nip(n8); null_1->nin(n2);
            null_1->nop(n3); null_1->non(gnd);

            r_3 = new sca_eln::sca_r("r_3", r);
            r_3->p(n3); r_3->n(n4);

            c_1 = new sca_eln::sca_c("c_1", c);
            c_1->p(n4); c_1->n(n5);



            null_2 = new sca_eln::sca_nullor("null_2");
```

```
        null_2->nip(gnd); null_2->nin(n4);
        null_2->nop(n5); null_2->non(gnd);

        r_4 = new sca_eln::sca_r("r_4", r);
        r_4->p(n5); r_4->n(n8);

        r_5 = new sca_eln::sca_r("r_5", r);
        r_5->p(n5); r_5->n(n6);

        c_2 = new sca_eln::sca_c("c_2", c);
        c_2->p(n6); c_2->n(n7);

        null_3 = new sca_eln::sca_nullor("null_3");
        null_3->nip(gnd); null_3->nin(n6);
        null_3->nop(n7); null_3->non(gnd);

        r_6 = new sca_eln::sca_r("r_6", r);
        r_6->p(n8); r_6->n(gnd);

        r_7 = new sca_eln::sca_r("r_7", r);
        r_7->p(n7); r_7->n(n2);

        r_L = new sca_eln::sca_r("r_L", r);
        r_L->p(n7); r_L->n(gnd);

        r_o = new sca_eln::sca_r("r_o", r/1000);
        r_o->p(n7); r_o->n(out);
    }
};
```

Annexe 2

Annexe 2.1

Ce code illustre comment créer le circuit électrique équivalent d'un transducteur électrodynamique

```
SC_MODULE(elec2meca_dynamic)
{
    sca_elec_port p_elec, n_elec, p_meca, n_meca;
    sca_eln::sca_l *ind;
    sca_eln::sca_gyrator *gyr;
    sca_elec_node n1;
    sc_in <double> val_ind, psi;
    double ratio;

    SC_CTOR(elec2meca_dynamic)
    {
        ratio = read(psi);
```

```
        ind = new sca_eln::sca_l("ind");
        ind->p(p_elec);
        ind->n(n0);
        ind->ctrl(val_ind);

        gyr = new sca_eln::sca_gyrator("gyr");
        gyr->p1(n0);
        gyr->n1(n_elec);
        gyr->p2(p_meca);
        gyr->n2(n_meca);
        gyr->g1 = -ratio;
        gyr->g2 = ratio;
    }
};
```

Annexe 2.2

Ce code présente le module `mod_eec`, soit le circuit électrique équiavelent d'un mode de vibrations du capteur de force.

```
SC_MODULE(mod_eec)
{
    sca_eln::sca_terminal p_elec, n_elec, p_meca, n_meca;
    sca_eln::sca_l *mass;
    sca_eln::sca_r *damp;
    sca_eln::sca_c *comp;
    sca_eln::sca_ideal_transformer *sigma, *phi;
    sca_eln::sca_node n1, n2, n3, n4;
    sca_eln::sca_node_ref gndm;

    double mass_v, damp_v, comp_v, sigma_v, phi_v;
    double mass_s, damp_s, comp_s, sigma_s, phi_s;


    mod_eec(sc_core::sc_module_name _n, double phi_v, double
mass_v, double damp_v, double comp_v, double sigma_v)
    {
        mass_s=mass_v;
        damp_s=damp_v;
        comp_s=comp_v;
        phi_s=phi_v;
        sigma_s=sigma_v;

        sigma=new sca_eln::sca_ideal_transformer("sigma");
        sigma->p1(p_elec);
        sigma->n1(n_elec);
```

```
        sigma->p2(n4);
        sigma->n2(gndm);
        sigma->ratio=sigma_s;

        comp=new sca_eln::sca_c("comp");
        comp->p(n3);
        comp->n(n4);
        comp->value=comp_s;

        damp=new sca_eln::sca_r("damp");
        damp->p(n2);
        damp->n(n3);
        damp->value=damp_s;

        mass=new sca_eln::sca_l("mass");
        mass->p(n1);
        mass->n(n2);
        mass->value=mass_s;

        phi=new sca_eln::sca_ideal_transformer("phi");
        phi->p1(n1);
        phi->n1(gndm);
        phi->p2(p_meca);
        phi->n2(n_meca);
        phi->ratio=phi_s;
    }
};
```

Annexe 2.3

Ce code présente le modèle haut niveau du capteur de force. Il permet de modéliser jusqu'à 5 modes de vibrations de la poutre.

```
SC_MODULE(force_sensor)
{
    sca_eln::sca_terminal p_elec, n_elec, p_meca, n_meca;

    int order_s;
    sca_util::sca_vector <double> phi_s;
    sca_util::sca_vector <double> sigma_s;
    sca_util::sca_vector <double> comp_s;
    sca_util::sca_vector <double> damp_s;
    sca_util::sca_vector <double> mass_s;

    mod_eec *mod[5];
```

```
     force_sensor(sc_core::sc_module_name   _n,   int   order_v,
sca_util::sca_vector   <double>   phi_v,   sca_util::sca_vector
<double>      sigma_v,      sca_util::sca_vector      <double>
comp_v,sca_util::sca_vector                                <double>
mass_v,sca_util::sca_vector <double> damp_v)
     {
          order_s = order_v;
          for(int i=0; i<order_s; i++)
          {
               mass_s(i)=mass_v(i);
               damp_s(i)=damp_v(i);
               comp_s(i)=comp_v(i);
               phi_s(i)=phi_v(i);
               sigma_s(i)=sigma_v(i);
          }

          for(int i=0; i<order_s; i++)
          {
               mod[i]=new  mod_eec("mod",  phi_s(i),  mass_s(i),
damp_s(i), comp_s(i), sigma_s(i));
               mod[i]->p_meca(p_meca);
               mod[i]->n_meca(n_meca);
               mod[i]->p_elec(p_elec);
               mod[i]->n_elec(n_elec);
          }
     }
};
```

Annexe 3

Nous présentons ici le code du redresseur double alternance triphasé.

```
SCA_TDF_MODULE (Red_tri)
{
     sca_tdf::sca_in <double> v1_s, v2_s, v3_s;
     sca_tdf::sca_ou <double> uout;
     SCA_CTOR(Red_tri){}
     void processing()
     {
          double v1, v2, v3, vout;
          double th;
          th = 1.4;
          v1 = v1_s.read();
          v2 = v2_s.read();
          v3 = v3_s.read();
          if (v1 > v2 && v1 > v3)
          {
```

```
                if (v2 > v3)
                     vout = v1 – v3;
                else
                     vout = v1 – v2;
           }
           if (v2 > v1 && v2 > v3)
           {
                if (v1 > v3)
                     vout = v2 – v3;
                else
                     vout = v2 – v1;
           }
           else
           {
                if (v1 > v2)
                     vout = v3 – v2;
                else
                     vout = v3 – v1;
           }
           uout.write(vout-th);
      }
};
```

Bibliographie

[Accellera 02] SystemC Analog/Mixed-signal Working Group (AMSWG). Accellera Systems Initiative, http://www.accellera.org/activities/committees/systemc-ams/$

[Accellera 09] Accellera. Verilog-AMS Language Reference Manual Version 2.3.1: Analog & Mixed-Signal Extensions to Verilog HDL. Accellera. 1370 Trancas Street, #163, Napa, CA 94558, USA, June 1, 2009.

[Accellera 11] SystemC AMS Day 2011 - Industry Adoption of the SystemC AMS Standard. Accellera Systems Initiative, May 2011. http://www.accellera.org/news/events/systemc_ams_day

[Accellera 12] SystemC AMS 2.0 Draft Standard. Accellera Systems Initiative, March 2012. http://www.accellera.org/activities/committees/systemc-ams/ams_2_draft_public_review/

[Agilent 04] Agilent technologies. ADS Ptolemy Simulation. 2004. http://cp.literature.agilent.com/litweb/pdf/ads2004a/pdf/ptolemy.pdf

[Alassir 06] M. Alassir, J. Denoulet, O. Romain & P. Garda. A SystemC AMS Model of an I2C Bus Controller. International Conference on Design and Test of Integrated Systems in Nanoscale Technology (DTIS 2006), pages 154-158, sept. 2006.

[Alassir 07] M. Alassir, J. Denoulet, O. Romain & P. Garda. Modeling I2C Communication Between SoCs with SystemC-AMS. IEEE International Symposium on Industrial Electronics (ISIE 2007), pages 1412-1417, June 2007.

[Al-Junaid 05] H. Al-Junaid & T. Kazmierski. Analogue and Mixed-Signal Extension to SystemC. IEEE Proceedings, Circuits, Devices and Systems, pages 682-690, dec. 2005.

[Al-Junaid 06] H. Al-Junaid, T. Kazmierski & L. Wang. SystemC-A Modeling of an Automotive Seating Vibration Isolation System. Forum on Design Languages (FDL 2006), pages 107-113, 2006.

[Aynsley 09] John Aynsley. SystemC versus SystemVerilog. *Doulos*, Feb. 2009. http://www.doulos.com/knowhow/video_gallery/#anchor3.

[Bakka 11] T. Bakka, H.R. Karimi, Wind Turbine Modeling Using The Bond Graph, IEEE International Symposium on Computer-Aided Control System Design (CACSD),Part of 2011 IEEE Multi-Conference on Systems and Control, Denver, CO, USA. September 28-30, 2011

[Balarin 03] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone & A. Savignolli-Vincentelli. Metropolis: An Integrated Electronic System Design Environnement. Computer, vol. 36, no. 4, pages 45-52, April 2003.

[Barnasconi 10] M. Barnasconi, C. Grimm, M. Damn, K. Enwich, M-M. Louërat, T. Mähne, F. Pecheux & A. Vachoux. SystemC AMS Extensions User's Guide. Open SystemC Initiative (OSCI). March 2008.

[Berman 05] V. Berman. A Tale of Two Languages: SystemC and SystemVerilog. Chip Design Magazine, July 2005.
http://chipdesignmag.com/display.php?articleId=116&issueId=11%09

[Biagetti 04] G. Biagetti, M. Caldari, M. Conti & S. Orcioni. Extending SystemC to Analog Modelling and Simulation. Languages for System Specification, pages 229-242. Springer US, 2004.

[Bjureus 01] P. Bjureus & A. Jantsch. Modeling of Mixed Control Dataflow Systems in MASCOT. IEEE Transactions on Very Large Scale Integration Systems (VLSI), vol. 9, no. 5, pages 6690-703, oct. 2001.

[Black 10] David C. Black et al. SystemC: From the Ground Up. 2nd ed. New York, Dordrecht, Heidelberg, London: Springer, 2010.

[Bondgraphs 13].Page web disponible à l'adresse url : http://bondgraph.org/software.html, consulté le 20 août 2013.

[Bonnerud 01] T.E. Bonnerud, B. Hernes & T. Ytterdal. A mixed-signal, functional level simulation framework based on SystemC for system-on-chip applications. IEEE Conference on Custom Integrated Circuits, 2001, pages 541-544.

[Bousquet 11a] L. Bousquet, F. Cenni, E. Simeu, SystemC AMS High-level Modeling of Linear Analog Blocks With Power Consumption Information, Latin American Test Workshop (LATW 2011), Mars 2011.

[Bousquet 11b] L. Bousquet, F. Cenni, E. Simeu, Inclusion of Power Consumption Information in High-level Modeling of Linear Analog Blocks, Journal of Low Power Electronics (JOLPE), décembre 2011.

[Bousquet 12] L. Bousquet, E. Simeu, « High-level Modeling of Power Consumption in Active Linear Analog Circuits », 22nd Great Lakes Symposium on Very Large Scale Integration (GLSVLSI 2012), Salt Lake City, Utah, May 2012.

[Bousquet 13] L. Bousquet, E. Simeu, « System-level Modeling of Electromechanical Devices With Energy Consumption », 7th International IEEE Systems Conference (Syscon 2013), Orlando, Florida, April 2013.

[Bowen 11] Matthew Bowen. Handel-C Language Reference Manual Version 2.1. Embedded Solutions Limited. url: http://www.pa.msu.edu/hep/d0/l2/Handel-C/Handel%20C.PDF (visited on 01/08/2011)

[Brooks 10] C. Brooks, E.A. Lee & S. Tripakis. Exploring Models of Computation with Ptolemy II. IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, pages 331-332, oct. 2010.

[CalcEdge] http://www.calculatoredge.com

[Caluwaerts 08] K. Caluwaerts, D. Galayko & P. Basset. SystemC-AMS Heterogeneous Modeling of a Capacitive Harvester of Vibration Energy. Beavioral Modeling and Simulation Workshop (BMAS), 2008.

[Cenni 09] F. Cenni, E. Simeu & S. Mir. Macro-modeling of Analog Blocks for SystemC-AMS Simulation : A Chemical Sensor Case Study. 17th IFIP International Conference on Very Large Scale Integration (VLSI-SoC), pages 211-214, October 2009.

[Cenni 11a] F. Cenni, S. Scotti & E. Simeu. SystemC AMS Behavioral Modeling of a CMOS Video Sensor. 19th IFIP /IEEE International Conference on Very Large Scale Integration (VLSI-SoC), pages 380-385, October 2011.

[Cenni 11b] F. Cenni, S. Scotti & E. Simeu. Behavioral Modeling of a CMOS Video Sensor Platform Using SystemC AMS/TLM. IEEE Forum on Design Languages (FDL 2011), September 2011.

[Cenni 11c] F. Cenni, S. Scotti & E. Simeu. A SystemC AMS/TLM Platform for CMOS Video Sensors. IEEE Conference on Design and Architectures for Signal and Image Processing (DASIP 2011), nov. 2011.

[Cenni 12] Fabio Cenni. Modélisation à haut niveau de systèmes hétérogènes, interfaçage analogique /numérique. Thèse de l'Université de Grenoble, April 2012.

[Chandrasekaran 10] S. Chandrasekaran. Verilog-AMS Integration with P1800 SV Standard. Requirements presentation. Accellera Verilog Analog Mixed-Signal Group, Feb. 10, 2010.

[Chung 75] Chung-Wen Ho, A. Ruehli, P. Brennan, The Modified Nodal Approach to Network Analysis, IEEE Transactions on Circuits and Systems, vol 22, no 6, pp 504-509, juin 1975.

[Clark 07] J.Clark, K.S.J Pister, Modeing, Simulation, and Verification of an Advanced Micromirror Using SUGAR, Journal of Microelectromechanical Systems, vol 16, no 6, December 2007.

[Damm 08a] M. Damm, J. Haase & C. Grimm. Co-Simulation of mixed HW/SW and Analog/RF systems at architectural level. Behavioral Modeling and Simulation Workshop, 2008. BMAS 2008. IEEE International, pp. 84 – 89, sept. 2008.

[Damm 08b] M. Damm, C. Grimm, J. Haas, A. Herrholz & W. Nebel. Connecting SystemC-AMS models with OSCI TLM 2.0 models using temporal decoupling. Specification, Verification and Design Languages, 2008. FDL 2008. Forum on, pp. 25 -30, sept. 2008.

[der Plas 02] G. Van der Plas, G. Gielen & W. Sansen. A Computer-Aided Design and Synthesis Environment for Analog Integrated Circuits. Boston, Dordrecht, London, Kluwer Academic Publishers, 2002.

[Dauphin 99] G. Dauphin-Tanguy. Les bond graphs et leurs application en mécatronique. 1999.

[Einwich 06] K. Einwich, J. Bastian, C. Clauss, U. Eichler & P. Schneider. SystemC-AMS Extension Library for Modeling Conservative Nonlinear Dynamic Systems. FDL, pages 113-119. ESCI, 2006.

[Einwich 10] K. Einwich, C. Grimm, W. Granig, G. Noessing, W.Scherr, S. Scotti, M. Barnasconi, G. Zucchelli & A. Vachoux. Requirements Specification for SystemC Analog Mixed Device (AMS) Extensions. *OSCI*, March 2010.

[Ferrari 12] A. Ferrari, L. Mangeruca, O. Ferrante, A. Mignogna, DesyreML: a SysML profile for heterogeneous embedded system, ERTS, February 2012,

[Ferro 11] Luca Ferro. Vérification de propriétés logico-temporelles de spécifications SystemC TLM. Thèse de l'Université de Grenoble, July 2011. http://tel.archives-ouvertes.fr/index.php?halsid=hs7mtscn20p28tgqlefomdd0f7&view_this_doc=tel-00633069&version=2

[Frevert 05] R. Frevert, J. Haase, and R. Jancke. Modeling and Simulation for RF System Design. Springer, Dec. 2005.

[Gajski 00] D. D. Gajski et al. SpecC: Specification Language and Methodology. Dordrecht, The Netherlands, Kluwer Academic Publishers, 2000.

[Gergaud 02] O.Gergaud. Modélisation énergétique et optimisation économique d'un système de production éolien et photovoltaïque couplé au réseau et associé à un accumulateur. Thèse de doctorat de l'Ecole Normale Supérieure de Cachan, Décembre 2002.

[Gerstlauer 01] A. Gerstlauer et al. System Design. A Practical Guide with SpecC. Dordrecht, The Netherlands, Kluwer Academic Publishers, 2001.

[Ghenassia 05] F. Ghenassia, ed. Transaction-Level Modeling with SystemC. TLM Concepts and Applications for Embedded Systems. Springer, 2005.

[Gielen 05] G.G.E. Gielen. CAD Tools for Embedded Analogue Circuits in Mixed-Signals Integrated Systems on Chip. IEEE Proceedings – Computers and Digital Techniques, vol. 152, no. 3, pages 317-332, May 2005.

[Goering 07] R. Goering. Magma Design claims first parallel fast Spice. EE Times, March 2007. http://www.eetindia.co.in/ART_8800456102_1800000_NP_e05da045.HTM

[Grimm 08] C. Grimm, M. Barnasconi, A. Vachoux & K. Einwich. An Introduction to Modeling Embedded Analog/Mixed-Signal Systems using SystemC-AMS Extensions. Open SystemC Initiative (OSCI), June 2008. http://publik.tuwien.ac.at/files/PubDat_171466.pdf

[Grötker 02] T. Grötker et al. System Design with SystemC. Springer, 2002.

[Hartmann 09] P.A. Hartmann, P. Reinkemeier, A. Rettberg & W. Nebel. Modelling Control Systems in SystemC AMS; Benefits and Limitations. IEEE International SoC Conference (SOCC 2009), pages 263-266, sept. 2009.

[Hartong 09] W. Hartong & S. Cranston. Real Valued Modeling for Mixed Signal Simulation. Cadence Design Systems, Jan. 2009. En ligne sur http://www.cadence.com/rl/Resources/application_notes/real_number_appNote.pdf

[Herrera 06] F. Herrera & E. Villar. A Framework for Embedded System Specification under Differents Models of Computation in SystemC. 43rd IEEE/ACM Design Automation Conference, pages 911-914, 2006.

[Herrera 07] F. Herrera, E. Villar & C. Grimm. A General Approach to the Interoperability of HetSC and SystemC-AMS. Forum on Specification Design Languages (FDL 2007), pages 101-110, sept. 2007.

[Hörmann 11] L. B. Hörmann, P. M. Glatz, C. Steger, and R. Weiss, A SystemC-AMS Simulation Environment for the Evaluation of Energy Harvesting Wireless Sensor Networks. Proceedings of the 2011 International Symposium on Performance Evaluation of Computer and Telecommunication Systems, pages 247 – 252, 2011.

[IEEE 04] IEEE Computer Society. IEC/IEEE Behavioural Languages – Part 4: Verilog Hardware Description Language (Adoption of IEEE Std 1364-2001) 1364-2001. IEEE. 2004.

[IEEE 05] IEEE Computer Society. 1800-2005 IEEE Standard for System Verilog—Unified Hardware Design, Specification, and Verification Language. 1800-2005. IEEE. 2005.

[IEEE 06] IEEE Computer Society. 1666-2005 IEEE Standard SystemC Language Reference Manual. 1666-2005. IEEE. Mar. 31, 2006.

[IEEE 07] IEEE Computer Society. 1076.1-2007 IEEE Standard VHDL Analog and Mixed-Signal Extensions. 1076.1-2007. IEEE. Nov. 15, 2007.

[IEEE 09a] IEEE Computer Society. 1076-2008 IEEE Standard VHDL Language Reference Manual. 1076-2008. IEEE. Jan. 26, 2009.

[IEEE 09b] IEEE Computer Society. 1800-2009 IEEE Standard for System Verilog—Unified Hardware Design, Specification, and Verification Language. 1800-2009. IEEE. 2009.

[Karnane 09] K. Karnane, G. Curtis & R. Goering. Solutions For Mixed-Signal SoC Verification. Cadence Design Systems, 2009. En ligne sur http://www.cadence.com/rl/Resources/white_papers/ms_soc_verification_wp.pdf

[Khanhua 11] S. Khankhua, M. W. Ashraf, S. Tayyab, N. Afzulpurkar & C. Punyasai. Simulation of MEMS based micro-gyroscope using CoventorWare. 8th Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI/CON 2011), pages 22-25, May 2011.

[Lauwers 00] E. Lauwers, G. Gielen. ACTIF : a High-level Power Estimation Tool for Analog Continuous-Time Filters; ICCAD 2000 , pp 193-196.

[Lauwers 02] E. Lauwers, G. Gielen. Power Estimation Methods for Analog Circuits for Architectural Exploration of Integrated Systems. IEEE Transactions on VLSI Systems. Vol 10 no 2, pp 155-162. April 2002

[Legrand 04] F. Legrand, Modélisation de Circuits Electrotechniques en Vue de Leur Simulation-Réalisation d’un Simulateur, janvier 2004.

[Lévêque 10] A. Lévêque, F. Pêcheux, M.-M. Louërat, H. Aboushady, M. Vasilevski. SystemC AMS models for low-power heterogeneous designs : Application to a WSN for the detection of seismic perturbations, Workshop on Ultra-Low Power Sensor Networks (WUPS), 2010.

[Lichiardopol 07] A-M. Lichiardopol, L'approche Bond Graph pour la Découverte Technologique, Thèse de doctorat de l'Ecole Centrale de Lille et de l'Université Politehnica de Bucarest, 2007.

[Litovski 97] V. Litovski, M. Zwolinski, VLSI Circuit Simulation and Optimization, Chapman & Hall, 1997.

[Malik 08] A. F. Malik, M. Shoaib, S. Naseem & S. Riaz. Modeling and Designing of RF MEMS switch using ANSYS. 4th Conference on Emerging Technologies (ICET 2008), pages 44-49, Oct. 2008.

[Markert 07] E. Markert, M. Dienel, G. Herrmann & U. Heinkel. SystemC-AMS Assisted Design of an Inertial Navigation System. IEEE Sensors Journal, vol. 7, no. 5, pages 770-777, May 2007.

[Massouri 10] A. Massouri, A. Lévêque, L. Clavier, M. Vasilevski, A. Kaiser & M.M. Louerat. Baseband Fading Channel Simulator for Inter-Vehicle Communication using SystemC-AMS. 2010 IEEE International Behavioral Modeling and Simulation Conference (BMAS), pp. 2.3, Sept. 2010.

[Mathaikutty 05] D. A. Mathaikutty et al. UMoC++: A C++-Based Multi-MoC Modeling Environment. Applications of Specification and Design Languages for SoCs. Selected papers from FDL'05. Springer, 2006, pp. 115–130.

[MATLAB 11] MATLAB. Matlab-The Language of Technical Computing. The Mathworks, 1984-2011.

[Mähne 06] T. Mähne et al. Creating Virtual Prototypes of Complex MEMS Transducers Using Reduced-Order Modelling Methods and VHDL-AMS. Applications of Specification and Design Languages for SoCs. Selected papers from FDL'05. Dordrecht, The Netherlands, Springer, 2006, pp. 135–153.

[Mähne 11] Torsten Mähne. Efficient Modelling and Simulation Methodology for the Design of Heterogeneous Mixed-Signal Systems on Chip. Thèse de l'École polytechnique fédérale de Lausanne (EPFL), April 2011. http://library.epfl.ch/theses/?nr=4993

[Meirovitch 75] L. Meirovitch, Elements of Vibration Analysis (New York: McGraw-Hill) 1975.

[Mentor 11] Mentor Graphics Corporation. Handel-C Synthesis Methodology. Mentor Graphics Corporation. 2009–2011. url: http://www.mentor.com/products/fpga/handel-c/ (visited on 01/08/2011)

[Mitea 11] O. Mitea, M. Meissner & L. Hedrich. Topology Synthesis of Analog circuits with Yield Optimization and Evaluation using Pareto Fronts. 19th IEEE/IFIP International Conference on VLSI and System-on-Chip (VLSI-SoC 2011), pages 78-81, Oct. 2011.

[Moser 99] E. Moser & W. Nebel. Case Study: System Model of Crane and Embedded Control. IEEE Computer Society, DATE, page 721, 1999.

[Müller 03] W. Müller, W. Rosenstiel, J.Ruf, eds. SystemC. Methodologies and Applications. Springer, 2003.

[Nathke 04] L. Nathke, V. Burkhay, L. Hedrich & E. Barke. Hierarchical Automatic Behavioral Model Generation of Nonlinear Analog Circuits Based on Nonlinear Symbolic Techniques. Design, Automation and Test in Europe Conference and Exhibition (DATE 2004), vol. 1, pages 442-447, feb. 2004.

[Orcioni 08] S. Orcioni, M. Ballicchia, G. Biagetti, R. d'Aparo & M. Conti. System Level Modelling of RF IC in SystemC-WMS. EURASIP Journal on Embedded Systems, vol. 2008, n°1, pp. 371768, 2008. http://jes.eurasipjournals.com/content/2008/1/371768

[OSCI 09] OSCI TLM Working Group. OSCI TLM-2.0 Language Reference Manual. Version 2.0.1. Open SystemC Initiative (OSCI). July 2009. url: http://www.systemc.org/members/download_files/check_file?agreement=tlm_2-0-1_090723 (visited on 01/11/2011).

[OSCI 10a] Open SystemC Inititiative (OSCI). Homepage of the Open SystemC Inititiative (OSCI). Open SystemC Initiative (OSCI). 2010. url: http://www.systemc.org/ (visited on 10/12/2010)

[OSCI 10b] OSCI. Open SystemC Initiative (OSCI) Standard SystemC-AMS Extensions Language Reference Manual. March 2010.

[O'Connor 06] I. O'Connor. Applications of Specfication and Design Languages for SoCs, UML/XML-Based Approach to Hierarchical AMS Synthesis, pages 205-225. Springer, 2006.

[O'Connor 07] I. O'Connor, B. Courtois, K. Chakrabarty, N. Delorme, M. Hampton & J. Hartung. Heterogeneous Systems on Chip and Systems in Package. Design, Automation and Test Conference (DATE 2007), April 2007.

[Patel 04] H. D. Patel and S. K. Shukla. SystemC Kernel Extensions for Heterogeneous System Modeling. The Netherlands, Kluwer Academic Publishers, 2004.

[Patel 05] H. D. Patel and S. K. Shukla. Towards a heterogeneous simulation kernel for system-level models: a SystemC kernel for synchronous data flow models. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 24 (8 Aug. 2005), pp. 1261–1271.

[Patel 08] H. D. Patel and S. K. Shukla. Ingredients for Successful System Level Design Methodology. Springer, 2008.

[Paugnat 11] F. Paugnat, L. Bousquet, K. Morin-Allory, L. Fesquet. A Performance Comparison Between the SystemC AMS Models of Computation, edaWorkshop, May 2011.

[Paugnat 12] F. Paugnat, L. Fesquet, K. Morin-Allory. Model of a Simple yet Effective Operational Amplifier. International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD). Sevilla, Spain, Sept 2012.

[Pêcheux 05] F. Pêcheux, C. Lallement, and A. Vachoux. VHDL-AMS and Verilog-AMS as alternative hardware description languages for efficient modeling of multidiscipline systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 24 (2005), pp. 204–225.

[Sander 04] I. Sander & A. Jantsch. System Modeling and Transformational Design Refinement in ForSyDe [formal system design]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 23, no. 1, pages 17-32, jan. 2004.

[Santarini 08] M. Santarini. Nascentric intros Fast Spice simulator for digital, AMS, analog designs. EDN, Feb. 2008. http://www.edn.com/design/integrated-circuit-design/4327967/Nascentric-intros-Fast-Spice-simulator-for-digital-AMS-analog-designs

[Simeu 99] E. Simeu, A. W. Peters, I. Rayane, Automatic Design of Optimal Concurrent Fault Detector for Linear Analog Systems, 29th Annual International Symposium on Fault-Tolerant Computing, pp. 184-191, 1999.

[TheWindPower 13] Page web disponible à l'adresse url :http://thewindpower.net/statistics_world_fr.php, consulté le 27 août.

[Tilmans 96] Harrie. A C Timans, Equivalent Circuit Representation of Electromechanical Transducers: I. Lumped-parameter Systems, Journal of Micromechanics and Microengineering, vol 6, no 1.

[Tilmans 97] Harrie. A C Timans, Equivalent Circuit Representation of Electromechanical Transducers: II. Distributed-parameter Systems, Journal of Micromechanics and Microengineering, vol 7, no 4.

[Uhle 10] T. Uhle & K. Einwich. A SystemC AMS Extension for the Simulation of Non-Linear Circuits. IEEE International SoC Conference (SOCC 2010), pages 193-198, sept. 2010.

[Vachoux 03] A. Vachoux, C. Grimm & K, Enwich. SystemC-AMS Requirements, Design Objectives and Rationale. Design, Automation and Test in Europe Conference and Exhibition (DATE 2003), pages 388-393, 2003.

[Vachoux 05] A. Vachoux, C. Grimm & K. Enwich. Extending SystemC to Support Mixed Discrete-Continuous System modeling and Simulation. IEEE International Symposium on Circuits and Systems (ISCAS 2005), pages 5166-5169 Vol. 5, May 2005.

[Vachoux 06] A. Vachoux. Applications of Specification and Design Languages for SoCs: Selected Papers from FDL 2005. SystemC-WMS: Mixed-Signal Simulation Based on Wave Exchanges. The ChDL series. Springer, 2006.

[Vasilevski 07a] M. Vasilevski, F. Pêcheux, H. Aboushady, L. de Lamarre. Modeling and Refining Heterogeneous Systems with SystemC-AMS: Application to WSN. BMAS IEEE International Behavioral Modeling and Simulation Conference, San Jose, CA, USA, 2007.

[Vasilevski 07b] M. Vasilevski, H. Aboushady, F. Pêcheux, L. de Lamarre. Modeling Wireless Sensor Network Nodes Using SystemC-AMS. ICM International Conference on Microelectronics, Cairo, Egypt, 2007.

[Vasilevski 08a] M. Vasilevski, N. Beilleau, H. Aboushady, F. Pêcheux. Efficient and Refined Modeling of Wireless Sensor Network Nodes Using SystemC-AMS. PRIME IEEE Conference on Ph.D. Research in MicroElectronics and Electronics, Istanbul, Turkey, 2008.

[Vasilevski 08b] M. Vasilevski, F. Pecheux, N. Beilleau, H. Aboushady & K. Einwich. "Modeling and Refining Heterogeneous Systems With SystemC-AMS: Application to WSN". Design, Automation and Test in Europe, 2008. DATE '08, pp. 134 -139, March 2008.

[Vasilevski 08c] M. Vasilevski, F. Pecheux, N. Beilleau, H. Aboushady & K. Einwich. "Modeling and Refining Heterogeneous Systems With SystemC-AMS: Application to WSN". Design, Automation and Test in Europe, 2008. DATE '08, pp. 134 -139, March 2008.

[Zaidi 10] Y. Zaidi, C. Grimm & J. Haase. On Mixed Abstraction, Languages and Simulation Approach to Refinement with SystemC AMS. EURASIP Journal of Embedded Systems, 2010.

Publications réalisées dans le cadre de la thèse

Publications dans des revues internationales à comité de lecture

[J1] L. Bousquet, F. Cenni, E. Simeu, « Inclusion of Power Consumption Information in High-level Modeling of Linear Analog Blocks », *Journal of Low Power Electronics (JOLPE)*, Vol 7, n 4, October 2011.

Publications dans des conférences internationales avec actes et comités de lecture

[C1] L. Bousquet, F. Cenni, E. Simeu, « SystemC AMS High-level Modeling of Linear Analog Blocks With Power Consumption Information », *12th IEEE Latin American Test Workshop (LATW 2011)*, Porto de Galinhas, Brazil, March 2011.

[C2] F. Paugnat, L. Bousquet, K. Morin-Allory, L. Fesquet, « A Performance Comparison Between the SystemC AMS Models of Computation », *edaWorkshop 2011*, Dresden, Germany, May 2011.

[C3] L. Bousquet, E. Simeu, « High-level Modeling of Power Consumption in Active Linear Analog Circuits », *22nd Great Lakes Symposium on Very Large Scale Integration (GLSVLSI 2012)*, Salt Lake City, Utah, May 2012.

[C4] L. Bousquet, E. Simeu, « System-level Modeling of Electromechanical Devices With Energy Consumption », *7th International IEEE Systems Conference (Syscon 2013)*, Orlando, Florida, April 2013.

Autres Communications

[N1] F. Paugnat, L. Bousquet, K. Morin-Allory, L. Fesquet, « Analog Design Abstraction evels and SystemC AMS Models of Computation », *SystemC AMS Day 2011: Industry Adoption of the SystemC AMS Standard*, Dresden, Germany, May 2011.

[N2] L. Bousquet, E. Simeu, « Including Power Consumption Information in SystemC AMS Modeling of Linear Analog Blocks at LSF MoC Level », *Colloque National du GDR SoC-SiP 2011*, Lyon, France, Juin 2011.

[N3] L. Bousquet, E. Simeu, « High-level Modeling of Linear Analog Blocks With Power Consumption Information », *Journées Scientifiques du Projet SEmba 2011*, Valence, France, October 2011.

[N4] L. Bousquet, « Energy Consumption Information in High-level Models », *ACM SIGDA/EDAA PhD Forum at DATE 2013*, Grenoble, France, March 2013.

Génération de modèles de haut niveau pour les systèmes hétérogènes et multiphysiques

Résumé

Les systèmes sur puce sont de plus en plus complexes : ils intègrent des parties numériques, des parties analogiques et des capteurs ou actionneurs. SystemC et son extension SystemC AMS permettent aujourd'hui de modéliser à haut niveau d'abstraction de tels systèmes. Ces outils constituent de véritables atouts dans une optique d'étude de faisabilité, d'exploration architecturale et de vérification du fonctionnement global des systèmes complexes hétérogènes et multiphysiques. En effet, les durées de simulation deviennent trop importantes pour envisager les simulations globales à bas niveau d'abstraction. De plus, les simulations basées sur l'utilisation conjointe de différents outils provoquent des problèmes de synchronisation. Les modèles de bas niveau, une fois crées par les spécialistes des différents domaines peuvent toutefois être abstraits afin de générer des modèles de haut niveau simulables sous SystemC/SystemC AMS en des temps de simulation réduits. Une analyse des modèles de calcul et des styles de modélisation possibles est d'abord présentée afin d'établir un lien avec les durées de simulation, ceci pour proposer un style de modélisation en fonction du niveau d'abstraction souhaité et de l'ampleur de la simulation à effectuer. Dans le cas des circuits analogiques linéaires, une méthode permettant de générer automatiquement des modèles de haut niveau d'abstraction à partir de modèles de bas niveau a été proposée. Afin d'évaluer très tôt dans le flot de conception la consommation d'un système, un moyen d'enrichir les modèles de haut niveau préalablement générés est présenté. L'attention a ensuite été portée sur la modélisation à haut niveau des systèmes multiphysiques. Deux méthodes y sont discutées : la méthode consistant à utiliser le circuit équivalent électrique puis la méthode basée sur les bond graphs. En particulier, nous proposons une méthode permettant de générer un modèle équivalent au bond graph à partir d'un modèle de bas niveau. Enfin, la modélisation d'un système éolien est étudiée afin d'illustrer les différents concepts présentés dans cette thèse.

Mots clés : SystemC AMS, niveaux d'abstraction, modèles de calcul, modélisation haut niveau, systèmes linéaires, systèmes hétérogènes, systèmes multiphysiques, bond graphs

Abstract

Systems on chip are more and more complex as they now embed not only digital and analog parts, but also sensors and actuators. SystemC and its extension SystemC AMS allow the high level modeling of such systems. These tools are efficient for feasibility study, architectural exploration and global verification of heterogeneous and multiphysics systems. At low level of abstraction, the simulation durations are too important. Moreover, synchronization problems appear when cosimulations are performed. It is possible to abstract the low level models that are developed by the specialists of the different domains to create high level models that can be simulated faster using SystemC/SystemC AMS. The models of computation and the modeling styles have been studied. A relation is shown between the modeling style, the model size and the simulation speed. A method that generates automatically the high level model of an analog linear circuit from its low level representation is proposed. Then, it is shown how to include in the high level model some information allowing the power consumption estimation. After that, the multiphysics systems modeling is studied. Two methods are discussed: firstly, the one that uses the electrical equivalent circuit, then the one based on the bond graph approach. It is shown how to generate a bond graph equivalent model from a low level representation. Finally, the modeling of a wind turbine system is discussed in order to illustrate the different concepts presented in this thesis.

Key words : SystemC AMS, abstraction levels, models of computation, high level modeling, linear systems, heterogeneous systems, multiphysics systems, bond graphs

ISBN 978-2-11-129187-4