Méthodologie de la conception de microsystèmes par la modélisation

Alexis Boegli

THÈSE SOUMISE À LA FACULTÉ DES SCIENCES
DE L'UNIVERSITÉ DE NEUCHÂTEL POUR L'OBTENTION
DU GRADE DE DOCTEUR ÈS SCIENCES

“La « Science » est un processus de découverte qui étudie les lois de fonctionnement du « Monde » qui nous gouverne. Personne n'invente rien, tout était là. Ces lois sont en activité avant même notre naissance. La science part à leur recherche en fonction des nécessités et des circonstances. Elle ne les crée pas. Voilà qui explique les multiples versions données de la Vérité au travers des siècles. Une fois découvertes, ces lois sont appliquées, expérimentées. Elles deviennent des règles qui aboutissent à créer et à promouvoir des activités nouvelles mises à profit jusqu'au moment où de nouvelles lois surgissent. Et cela se nomme la Recherche.”

Dr. J. Fontaine

IMPRIMATUR POUR LA THESE

Méthodologie de la conception de microsystèmes par la modélisation

de M. Alexis Boegli

UNIVERSITE DE NEUCHATEL

FACULTE DES SCIENCES

La Faculté des sciences de l'Université de Neuchâtel sur le rapport des membres du jury,

MM. F. Pellandini (directeur de thèse),
N. de Rooij, H.-P. Amann (Neuchâtel) et
A. Vachoux (Xemics SA, Neuchâtel)

autorise l'impression de la présente thèse.

Neuchâtel, le 13 février 2001

Le doyen:

J.-P. Derendinger

Résumé

Les systèmes de la microtechnique, appelés microsystèmes, sont généralement composés d'éléments micro-usinés (microstructures) et électroniques. Le marché de ces systèmes est actuellement en pleine expansion. Il est donc important de réagir rapidement à cette demande en réduisant les temps de développement et de mise au point. Mais cette réduction se heurte à la pluridisciplinarité des microsystèmes et donc au contrôle du développement et de la mise au point des différentes parties composant les microsystèmes.

Ce travail de thèse présente une méthode permettant de réaliser des microsystèmes avec un grand contrôle. Il s'agit d'une méthode de conception de type descendante appelée "approche MicroSIM". Cette méthode est basée sur la modélisation et s'appuie sur un environnement de conception assistée par ordinateur (CAO). Ce travail identifie et énumère les principales contraintes pesant sur cet environnement. Il énumère également un certain nombre de points auxquels il faut faire attention lors de l'élaboration de nouveaux modèles.

Afin de tester l'approche, un environnement de développement professionnel destiné à l'électronique a été choisi et utilisé. Pour qu'il soit utilisable dans le contexte des microsystèmes, un design-kit a été développé. Il s'agit essentiellement d'une bibliothèque de modèles appelée "GLOBS Microsystem Design Kit".

Finalement, les exemples de réalisations présentés dans ce travail démontrent que la méthode permet effectivement de réaliser des microsystèmes avec un grand contrôle.

Zusammenfassung

In dieser Doktorarbeit interessieren wir uns für neuzeitliche mikrotechnische Systeme, sogenannte Mikrosysteme. Diese bestehen normalerweise aus mikro-maschinell hergestellten Komponenten (Mikrostrukturen) und Elektronik. Der Markt für diese Systeme befindet sich derzeit in starker Expansion. Um dieser Nachfrage wirksam zu begegnen, müssen die Entwicklungszeit und die Testphase verkürzt werden. Diese Zeitoptimierung scheitert aber daran, dass Mikrosysteme aus Komponenten verschiedenster Fachgebiete bestehen. Aus diesem Grund sind Entwicklung und Test das zentrale Problem.

Wir stellen eine absteigende Methode vor, mit welcher Mikrosysteme, sogenannte MicroSIM, unter strenger Kontrolle entwickelt werden können. MicroSIM basiert auf der Herstellung von Modellen und auf deren Simulation in einer CAD-Umgebung (Computer-Aided-Design). Wir haben auch die Voraussetzungen analysiert und aufgelistet, denen die CAD-Umgebung genügen muss und welche Punkte bei der Entwicklung neuer Modelle zu respektieren sind.

Als Testmethode wurde eine professionelle CAD-Umgebung für Elektronik ausgewählt. Diese wurde mit einem Design-kit, dem sogenannten GLOBS Microsystems Design-Kit, das hauptsächlich aus Modellen besteht, den Bedürfnissen der Mikrosysteme angepasst.

Als Resultat beweisen die in dieser Arbeit vorgestellten Prototypen dass die Entwicklung von Mikrosystemen unter strenger Kontrolle mit der vorgestellten Methode möglich ist.

Abstract

The following work concentrates mainly on modern microtechnology systems, more commonly referred to as microsystems. These microsystems are generally composed of micromachined components (the microstructures) and electronics. Microsystems are a rapidly growing field, with significant potential for the future. As a result of this rapid progression, minimizing the time required for development and test processes becomes a challenging task. Furthermore, the problem of bringing together the many disciplines of the microsystem field arises. Therefore, development and debugging processes of the different microsystem parts are central to the problem.

In this thesis, the presentation is a global "top-down" approach for microsystems. Referred to as MicroSIM, the approach should allow a greater development control of microsystems. MicroSIM is based on modeling and simulation in a complex Computer-Aided-Design (CAD) environment. This work lists the constraints the CAD environment has to bear. It also enumerates some points to consider when developing new models.

To test the fore-mentioned approach, a professional electronic-dedicated CAD environment was selected. To use it in the microsystem context, a design-kit was developed. This design-kit, referred to as GLOBS Microsystem Design-Kit, mainly contains models usable in the selected CAD environment.

Finally, the examples which are presented, give proof that, following the MicroSIM approach, the microsystems can be developed with significant control.

Table des matières

Chapitre 1

Introduction

La recherche présentée dans ce travail de thèse est consacrée à une méthode de conception de microsystèmes. Cette méthode est basée sur la modélisation et la simulation globale suivant un mode de conception descendant.

1.1 Motivations

Le développement des microsystèmes est considéré aux USA, en Europe et au Japon comme stratégique et prioritaire. Comme le montre une étude de l'*ARPA* (Advanced Research Project Agency), le marché des microsystèmes est en pleine expansion (table 1-1). Il est donc important de réagir rapidement à cette demande et de réduire les temps de développement et de mise au point.

En Suisse, les plus hautes autorités ont également perçu l'importance des microsystèmes. Ainsi le 12 mai 1999, le Conseil fédéral a approuvé le mandat de prestations du Conseil des *EPF* (écoles polytechniques fédérales). Selon ce mandat, le Conseil des EPF s'engage notamment à encourager de manière prioritaire les microsystèmes et les nanotechnologies. Notons, au passage, que ce travail a été en partie financé par le projet MicroSIM du programme prioritaire Suisse *MINAST* (Micro & Nano System Technology 1997-1999).

Les microsystèmes sont généralement composés d'éléments micro-usinés et électroniques. Les éléments micro-usinés peuvent être électromécaniques, électro-optiques, etc. Il s'agit de ce que nous appelons les microstructures. L'électronique est généralement formée de parties analogiques et numériques. Elle peut également contenir des composants programmables tels que des microcontrôleurs.

Types de dispositifs	***1994***	***1996***	***2000****
Capteurs de pression	0.3	0.6	3.5
Interrupteurs optiques	--	0.1	3.9
Capteurs inertiels	0.1	0.3	2.8
Régulateurs Fluidiques	0.5	0.8	2.7
Mémoires	--	0.1	0.8
Autres	0.1	0.2	1.3
Total (milliard de $)	***1***	***2.1***	***15***

Table 1-1. Résultats d'une étude de marché effectuée par l'ARPA [Arp99] [Sen98a] [Mar97]. * Prévision pour l'année 2000.

Nous le constatons, la réduction des temps de développement et de mise au point se heurte à la pluridisciplinarité des microsystèmes. Idéalement, il faut disposer d'outils permettant de spécifier, de concevoir et de contrôler les différentes parties d'un microsystème avant que la réalisation détaillée ne démarre. Le développement parallèle des divers éléments est alors possible. Lors de l'assemblage, la probabilité d'être confronté à de mauvaises surprises, coûteuses en temps et en argent, est ainsi fortement réduite.

Depuis l'invention du transistor en 1948 et des premiers circuits intégrés dans les années 60, les processus de fabrication n'ont cessé d'évoluer. Actuellement, les technologies les plus avancées autorisent une définition inférieure à 0.2 µm alors que les plus gros

processeurs comptent plus de 10 millions de transistors. On conçoit aisément qu'il n'est simplement pas possible de créer de tels circuits sans l'aide d'outils de conception assistée par ordinateur (*CAO* ou *CAD*: Computer-Aided-Design). On conçoit également qu'il n'est pas possible de tester tous les transistors, voir tous les blocs fonctionnels d'un circuit après intégration. La simulation avant réalisation joue donc un rôle-clé dans la conception de circuits intégrés.

L'histoire des outils CAO est bien évidemment liée au développement des ordinateurs. En électronique, les premiers simulateurs datent du milieu des années 70. Il s'agit de simulateurs de temps continus créés pour analyser des schémas d'électronique analogique. Plus tard, c'est l'électronique numérique qui connaît un fort développement et c'est en 1987 que voit le jour le premier langage normalisé permettant de décrire du hardware numérique. Basés sur de tels langages, c'est dans les années 90 que les premiers outils de synthèse automatique font leur apparition. Finalement, ce n'est que très récemment (1999) que le premier langage de description de hardware numérique et analogique fut accepté comme norme.

Les premières microstructures fabriquées en utilisant des processus de fabrication des semi-conducteurs datent des années 70. Ce sont elles qui ont ouvert la voie des microsystèmes. Pour développer les microstructures, les outils de CAO sont également très importants. Les bases de la méthode des éléments finis, communément utilisée pour analyser une microstructure, datent de 1943 déjà. Mais ce n'est que dans les années 60 qu'apparaissent les premiers programmes informatiques basés sur cette méthode. Depuis, ces outils ont considérablement évolué et touchent à de nombreux domaines.

Ainsi, pour développer de l'électronique analogique ou numérique et des microstructures, des outils de CAO existent et sont communément utilisés. Il existe cependant de grandes différences entre ces outils. Par conséquent, ils ne sont pas utilisés pour développer des microsystèmes complets, mais des parties de microsystèmes uniquement.

1.2 Portée du travail

Nous venons de le voir, le concepteur de microsystèmes est confronté au problème de la pluridisciplinarité des microsystèmes. Pour réaliser un microsystème, la méthode qui consiste à le diviser en sous-systèmes puis à développer les sous-systèmes séparément pour finalement les réunir n'est pas satisfaisante. Une méthode basée sur des outils CAO est donc nécessaire aux concepteurs de microsystèmes de manière à leur offrir un grand contrôle du développement.

Dans ce travail, nous allons nous intéresser aux différentes méthodes de conception. Plus précisément, nous nous intéresserons aux différents processus de conception dans le but de voir quels sont ceux généralement utilisés pour concevoir des microstructures, de l'électronique analogique, de l'électronique numérique et du software. Ensuite nous proposerons une méthode de conception globale pour réaliser des microsystèmes. Cette méthode de conception débute avec un cahier des charges et se termine lorsque des prototypes d'un microsystème sont construits et sont prêts à être testés. Nous pourrons alors établir les contraintes pesant sur l'environnement de CAO. Nous regarderons si de tels environnements existent ou si certains environnements existants peuvent, moyennant quelques améliorations, faire l'affaire. Nous serons alors en mesure de montrer comment différents outils CAO existants peuvent être utilisés pour le design de microsystèmes dans le cadre d'une approche globale. Finalement nous essayerons de distinguer les limites de la méthode et certains pièges à éviter.

Il est à remarquer que ce travail ne porte pas sur le développement de microstructures ou d'électronique mais bien sur les méthodes de conception. Nous venons de voir que les microsystèmes sont composés d'électronique et de microstructures. Ils peuvent également contenir une (et même plusieurs) partie software. Dans ce travail, nous ne nous occuperons pas des méthodes visant à séparer la fonctionnalité désirée en hardware et software.

Les microsystèmes touchant généralement à différentes disciplines, la conception et la réalisation de ceux-ci font donc appel à

des spécialistes. Dans le cadre de ce travail, nous avons pu bénéficier du soutien et de la collaboration d'autres groupes de recherches et d'entreprises industrielles privées: IMT SAMLAB, EPFL DMX-LC, EPFL DMT-IMS, ETA SA, Ensyma SA, Haenni SA, Analog Microelectronics GmbH, Thermo Orion, Valtronic SA (cf. références § 1.5).

Pour réaliser les exemples présentés dans ce travail, nous avons bénéficié des connaissances de spécialistes: physiciens, chimistes, électroniciens, microthechnicien et informaticiens. Ainsi les modèles relatifs à l'amortissement de vibrations (§ 4.3.3) ont joui de la collaboration de D. Daudet, E. Colla et B. Waarsing. Les modèles des différentes parties du micromoteur ultrasonique (§ 4.3.4) ont été développés avec G.-A. Racine et L. Dellman. L'électronique autorisant la linéarisation de mesures de pression (§ 5.1) a été réalisée par Ch. Berthoud, R. Riem-Vis A. Heubi et U. Seger. L'étude des capteurs, le développement de l'électronique et la programmation du système permettant de mesurer le pH, la conductivité et la température de liquides (§ 5.2) ont été réalisés avec la collaboration de P. van der Wal, S. Pata, B. van der Schoot et H. Delabre.

Pour finir, précisons que ce travail a été financé par les projets: MicroSIM MINAST, Thermo Orion, CTI-3190, FUSE 22.920 et FUSE 26.907.

1.3 Organisation du rapport

- Dans le chapitre 2, nous commençons par fixer la terminologie utilisée dans ce travail. Ensuite nous présentons les principaux modes de conception ainsi que la solution adoptée. Il s'agit de ce que nous appelons "l'approche MicroSIM".
- Le chapitre 3 est consacré aux langages de modélisation et aux simulateurs existants. Nous en soulignerons les différences afin d'effectuer un choix compatible avec l'approche MicroSIM. Nous verrons que, pour que cette approche soit applicable, il faut que des design-kits (sortes de bibliothèques) existent.

- La première partie du chapitre 4 présente un diagramme permettant d'énumérer les différents points de vue et niveaux d'abstraction d'un sous-système formant un microsystème. La suite du chapitre présente le design-kit que nous avons créé dans le but d'appliquer l'approche MicroSIM, ainsi que des exemples de modèles de sous-systèmes illustrant les difficultés rencontrées lors de leur élaboration.
- Le chapitre 5 est consacré à deux exemples de microsystèmes réalisés en suivant certaines étapes de l'approche MicroSIM. Ces exemples permettent de se rendre compte des avantages et des inconvénients de la méthode.
- Finalement le dernier chapitre, le chapitre 6, est consacré à la conclusion.

1.4 Principales contributions

Dans ce travail, nous présentons une approche permettant de réaliser des microsystèmes avec un grand contrôle. Il s'agit d'une approche de type descendante appelée "approche MicroSIM".

Cette méthode est basée sur la modélisation et s'appuie sur un environnement de CAO. Ce travail identifie et énumère les principales contraintes pesant sur cet environnement. Il énumère également un certain nombre de points auxquels il faut faire attention lors de l'élaboration de nouveaux modèles.

Afin de tester l'approche, nous avons choisi d'utiliser un environnement destiné à l'électronique. Pour qu'il soit utilisable dans le contexte des microsystèmes, nous avons développé un design-kit. Il s'agit essentiellement d'une bibliothèque de modèles.

1.5 Références

[AMS99] Analog Microelectronics GmbH, http://www.analogmicro.de

[Arp99] *"According to a market study done by the Advanced Research Projects Agency (ARPA), it is estimated that the MEMS market*

will grow from $1.6 to $14 billion by the year 2000", http://www.ultratech.com/content_pages/product/misc/microm.html

[DMT99] Ecole Polytechnique Fédérale de Lausanne, Département de Microtechnique, Institut de Microsystèmes, EPFL DMT-IMS, http://dmtwww.epfl.ch

[DMX99] Ecole Polytechnique Fédérale de Lausanne, Département des Matériaux, Laboratoire de Céramique, EPFL DMX-LC, http://dmxwww.epfl.ch

[Ens99] Ensyma SA, Jaquet-Droz 1, CH-2000 Neuchâtel

[ETA99] ETA SA Fabriques d'Ebauches, http://www.eta.ch

[Hae99] Haenni SA, http://www.haenni-instr.com

[Mar97] S.Marshall, *"MEMS market data: Another case of - sorry - wrong number?"*, Micromachined devices, 1997, p.6

[Ori99] Thermo Orion, http://www.orionres.com

[SAM99] Université de Neuchâtel, Institut de microtechnique, Sensors, Actuators and Microsystems Laboratory, IMT SAMLAB, http://www-samlab.unine.ch

[Sen98a] S.D.Senturia, *"Design of Microelectromechanical Devices, A Short Course"*, cours organisé par la FSRM et donné par le professeur S. D. Senturia à Neuchâtel en juin 1998

[Val99] Valtronic SA, http://www.valtronic.com

Chapitre 2

Approche *MicroSIM*

Les microsystèmes peuvent être composés de sous-systèmes électroniques (analogiques et numériques), électromécaniques, électro-optiques, etc. Le concepteur de microsystèmes est confronté au problème de la complexité et de l'interdisciplinarité des sous-systèmes. Les couplages entre les sous-systèmes sont parfois importants. Une méthode basée sur des outils de CAO est nécessaire aux concepteurs de microsystèmes: il s'agit ici de l'approche MicroSIM. Cette méthode utilise la modélisation comportementale et s'appuie sur des bibliothèques d'éléments nommés design-kit. L'approche MicroSIM est de type "descendante", et a pour principal avantage d'offrir un grand contrôle du développement d'un microsystème.

Dans ce chapitre, nous commençons par introduire la terminologie liée à la conception de microsystèmes. Ensuite nous verrons ce que sont la conception descendante et ascendante, et comment ces deux approches sont adaptées à la réalisation de sous-systèmes. Nous constaterons qu'il n'est pas satisfaisant de réaliser des sous-systèmes séparément pour finalement les assembler en un microsystème. Pour terminer, nous présenterons la solution que nous avons adoptée, "l'approche MicroSIM".

2.1 Terminologie

Dans ce travail nous nous intéressons aux systèmes de la microtechnique, appelés microsystèmes. Ce terme de microsystème est un terme utilisé en Europe, alors qu'aux USA, les spécialistes parlent de microsystèmes électromécaniques (*MEMS*: Micro-Electro-Mechanical Systems). Bien que formellement les MEMS soient un sous-ensemble des microsystèmes, le terme MEMS est utilisé par exemple pour désigner des microsystèmes optoélectroniques.

En consultant des publications relatives au domaine [Ama99] [Boe96a] [Pop95], nous constatons que la définition d'un microsystème (ou d'un MEMS) varie de cas en cas. Pour certains auteurs, une *microstructure* (c'est-à-dire un transducteur de petites dimensions) constitue déjà un microsystème. Pour d'autres, il faut que la microstructure soit reliée à un minimum d'électronique analogique et/ou numérique.

La définition d'un microsystème que nous avons retenue est la suivante: un *microsystème* est un système de petite dimension (jusqu'à quelques centimètres), remplissant une fonction intelligente, c'est-à-dire possédant des moyens propres de traitement et une certaine autonomie de fonctionnement.

L'approche MicroSIM que nous présentons dans ce chapitre a pour but de promouvoir une méthode pour réaliser des microsystèmes ainsi qu'un environnement de CAO pour microsystèmes. Ainsi, lorsque le fonctionnement d'un microsystème est simulé, nous parlons de *simulation système*.

Les modèles représentant les différentes parties du système peuvent décrire la réalité avec plus ou moins de détails. Un modèle contenant peu de détails est appelé: *modèle haut-niveau* (modèle ayant un niveau d'abstraction élevé). Inversement, un modèle décrivant la réalité avec beaucoup de détails, est appelé *modèle bas-niveau*. Lorsque des modèles haut-niveau et bas-niveau sont connectés et simulés ensemble, nous parlons de *simulation multi-niveaux*.

Les modèles de microstructures décrivent le comportement de transducteurs qui sont par exemple: électromécaniques, optoélectroniques, etc. Leurs signaux d'entrées et de sorties sont de nature différente (électrique, mécanique, etc.). Lorsque des signaux de différentes natures sont impliqués dans une simulation fonctionnelle, nous parlons de *simulation multi-natures*.

En électronique, il faut distinguer le monde numérique du monde analogique. Lorsque des modèles analogiques et numériques sont connectés et simulés ensemble, nous parlons de *simulations multi-modes*.

Selon notre définition, un microsystème remplit une fonction intelligente. Cette fonction peut prendre la forme d'un logiciel (assembleur, C, C++, etc.). Dans un tel cas, il peut être utile de simuler le logiciel (SoftWare*: SW*) avec son électronique (HardWare: *HW*, par exemple un microprocesseur ou un microcontrôleur). Nous parlons alors de *co-simulation HW-SW*.

Les procédures de tests d'un microsystème sont extrêmement importantes. C'est pourquoi, lorsque l'on en réalise un, on le connecte à un *banc de tests*. En simulation, il est également possible de soumettre un microsystème à un modèle de banc de tests. Lorsqu'un microsystème est simulé avec un banc de tests, nous parlons de *simulation globale*.

2.2 Problèmes et motivations

Pour certains spécialistes des microstructures, une microstructure est déjà un microsystème. Selon notre définition (§ 2.1), une microstructure seule n'est pas un microsystème dans la mesure où une unité d'acquisition et de traitement du signal est nécessaire à son fonctionnement. Ce problème de définition est révélateur d'une difficulté. Il s'agit de l'interdisciplinarité à laquelle sont confrontés les concepteurs de microsystèmes. Conformément à notre définition, un microsystème peut se composer (fig. 2-1) des *sous-systèmes* suivants:

1. Acquisition d'informations (capteurs).

2. Traitement du signal (électronique analogique et numérique + SW).
3. Partie pouvant entreprendre des actions (actionneurs).
4. Alimentation (électronique + piles).

La figure 2-1 nous montre à quel point les microsystèmes sont hétérogènes. La réalisation d'un microsystème touche à de multiples sciences: mécanique, physique, chimie et informatique.

En cherchant à classer les sous-systèmes d'un microsystème en fonction des méthodes de conception et développement mises en oeuvre pour les réaliser, nous obtenons trois groupes: microstructures, électronique et SW. En séparant l'électronique analogique de l'électronique numérique, nous obtenons des sous-groupes (fig. 2-2).

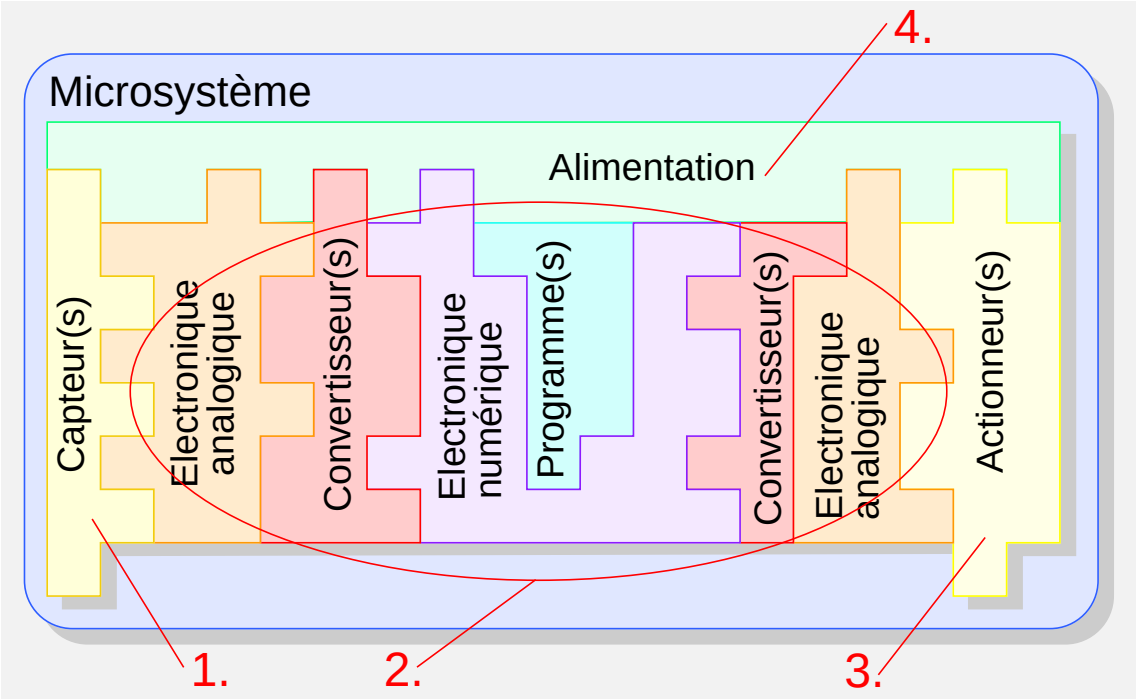


Figure 2-1. Sous-systèmes pouvant composer un microsystème: (1) acquisition d'information, (2) traitement du signal, (3) partie pouvant entreprendre des actions et (4) alimentation.

Lorsque nous parlons de conception de systèmes, de microsystèmes ou de sous-systèmes appartenant à l'un des groupes, nous entendons parler de *conception descendante* ou *ascendante*

[Mos96]. Dans ce qui suit, nous allons brièvement présenter ces deux approches. Ensuite nous verrons comment on peut concevoir séparément de l'électronique numérique, de l'électronique analogique et des microstructures. Finalement nous discuterons des problèmes rencontrés lorsque nous réalisons un microsystème.

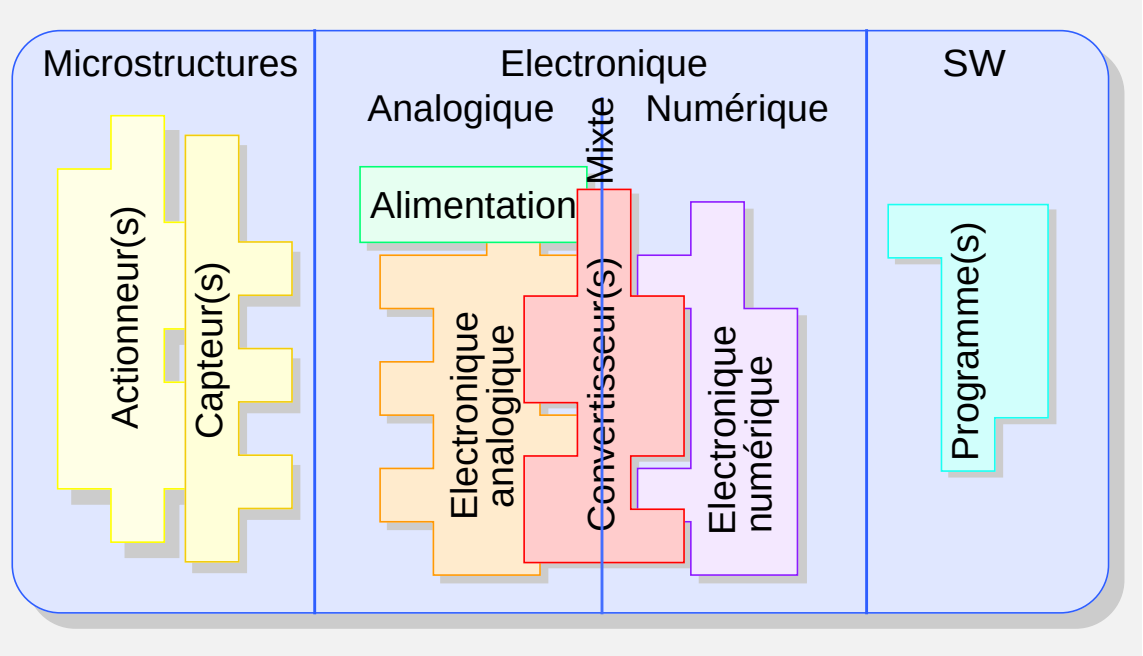


Figure 2-2. Groupement des sous-systèmes formant un microsystème en fonction des méthodes de conception.

2.2.1 Conception descendante et ascendante

Imaginons que nous devions concevoir un *système*, comme par exemple un des sous-systèmes d'un microsystème. Le point de départ de la conception descendante (fig. 2-3) est un cahier des charges contenant les spécifications du système ou du sous-système.

Après étude de différentes décompositions, l'une d'elles est adoptée. Cette opération de décomposition est répétée hiérarchiquement jusqu'à ce que des *primitives* soient atteintes. Dans le cas de l'électronique analogique, de l'électronique numérique et du SW, les primitives sont, respectivement:

- Le transistor, la résistance, l'inductance et la capacité.

- Les portes logiques.
- Les mots binaires à placer en mémoire RAM ou ROM.

Lors de chaque décomposition, les éléments sont ajustés de manière à ce que les spécifications de départ soient satisfaites. Cette opération de décomposition et d'ajustement est appelée *décomposition*. Le résultat de chaque décomposition doit être vérifié. La simulation est une façon d'effectuer cette opération de *vérification*.

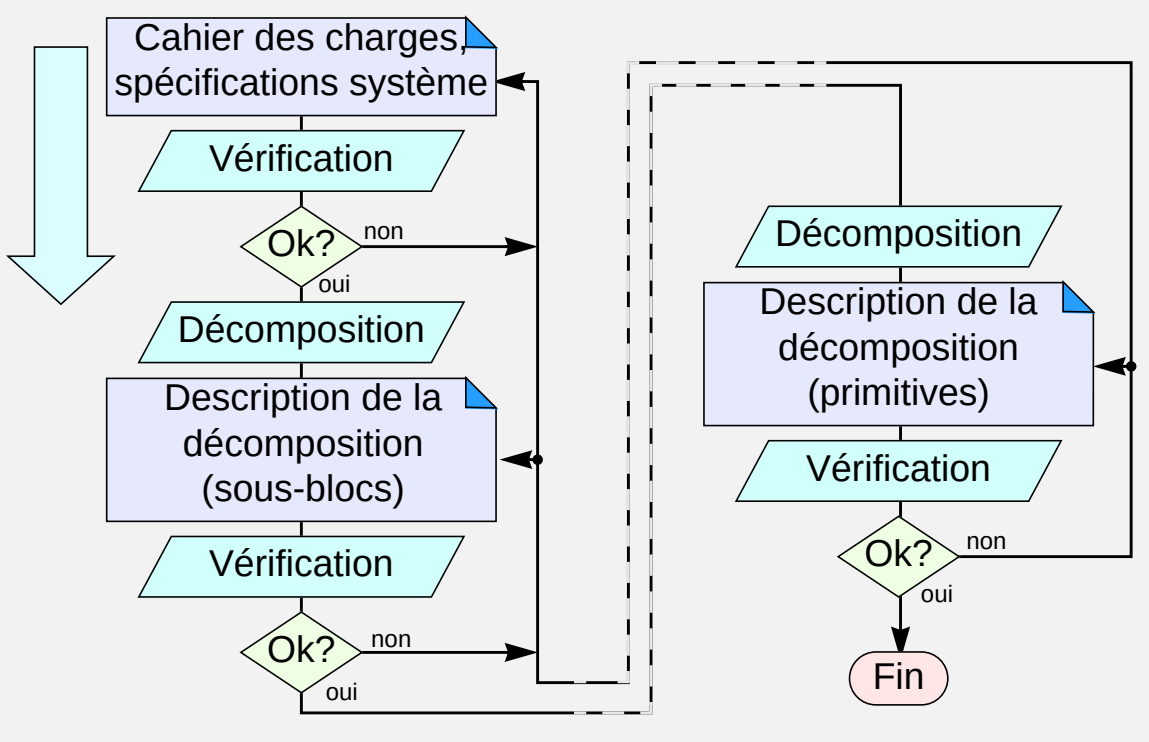


Figure 2-3. Conception descendante.

Dans le cas des microstructures, les primitives sont plus difficiles à définir. Certains travaux [Kar96] visent à réaliser des bibliothèques de microstructures pouvant être paramétrées. Dans ce cas ce sont les microstructures elles-mêmes qui sont les primitives.

La conception ascendante (fig. 2-4) est l'inverse de la conception descendante. Dans la conception ascendante, des éléments de base connus (les primitives de l'approche descendante) sont utilisés pour construire des blocs plus complexes. Ces blocs peuvent

ensuite être utilisés pour construire de nouveaux blocs encore plus complexes. Cette procédure est répétée jusqu'à ce que la fonctionnalité finale soit obtenue. L'opération d'assemblage de blocs visant à obtenir un bloc plus complexe est nommée *composition*. Après chaque composition, une opération de *caractérisation* est nécessaire pour vérifier si la fonctionnalité obtenue correspond à celle souhaitée.

Le principal avantage de la conception descendante, est le fait qu'on puisse vérifier les spécifications de départ après chaque décomposition. De plus, cette approche est couramment utilisée lors de la réalisation de circuits électroniques numérique. La conception ascendante ne présente pas le même avantage. Après une composition intermédiaire, ce ne sont que des parties du système final qui sont caractérisées. Ce n'est qu'après la dernière composition que l'on peut caractériser le système final.

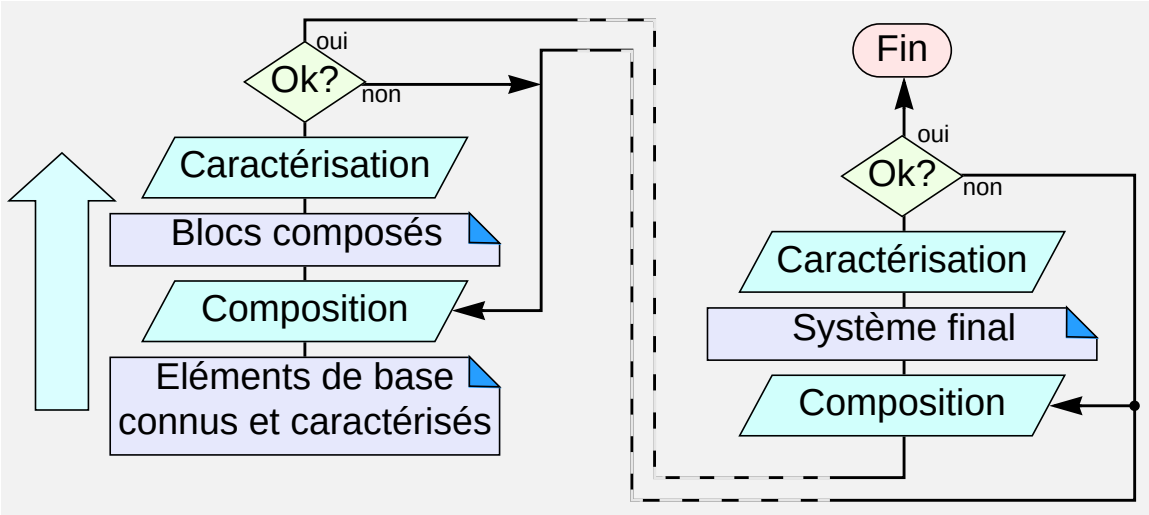


Figure 2-4. Conception ascendante.

Il est intéressant de noter que pour créer des systèmes relativement simples, tels qu'une collection de primitives indispensable à la conception descendante, c'est l'approche ascendante qui est utilisée. Inversement, nous constatons que c'est généralement l'approche descendante qui est choisie lorsque des systèmes complexes sont conçus.

Dans les paragraphes suivants, nous allons voir comment la conception descendante est adaptée à la réalisation d'électronique

numérique et analogique. Nous verrons ensuite quelle approche est adoptée lors de la réalisation de microstructures.

2.2.2 Conception descendante d'électronique numérique

Le point de départ de la conception descendante d'électronique numérique (fig. 2-5) est un cahier des charges contenant les spécifications du circuit. Après vérification, une première étape de décomposition(s). Lors de cette étape, un langage de description du HW (Hardware Description Language: *HDL*) tel que *VHDL* (*VHSIC* HDL, *VHSIC:* Very High Speed Integrated Circuit) est utilisé.

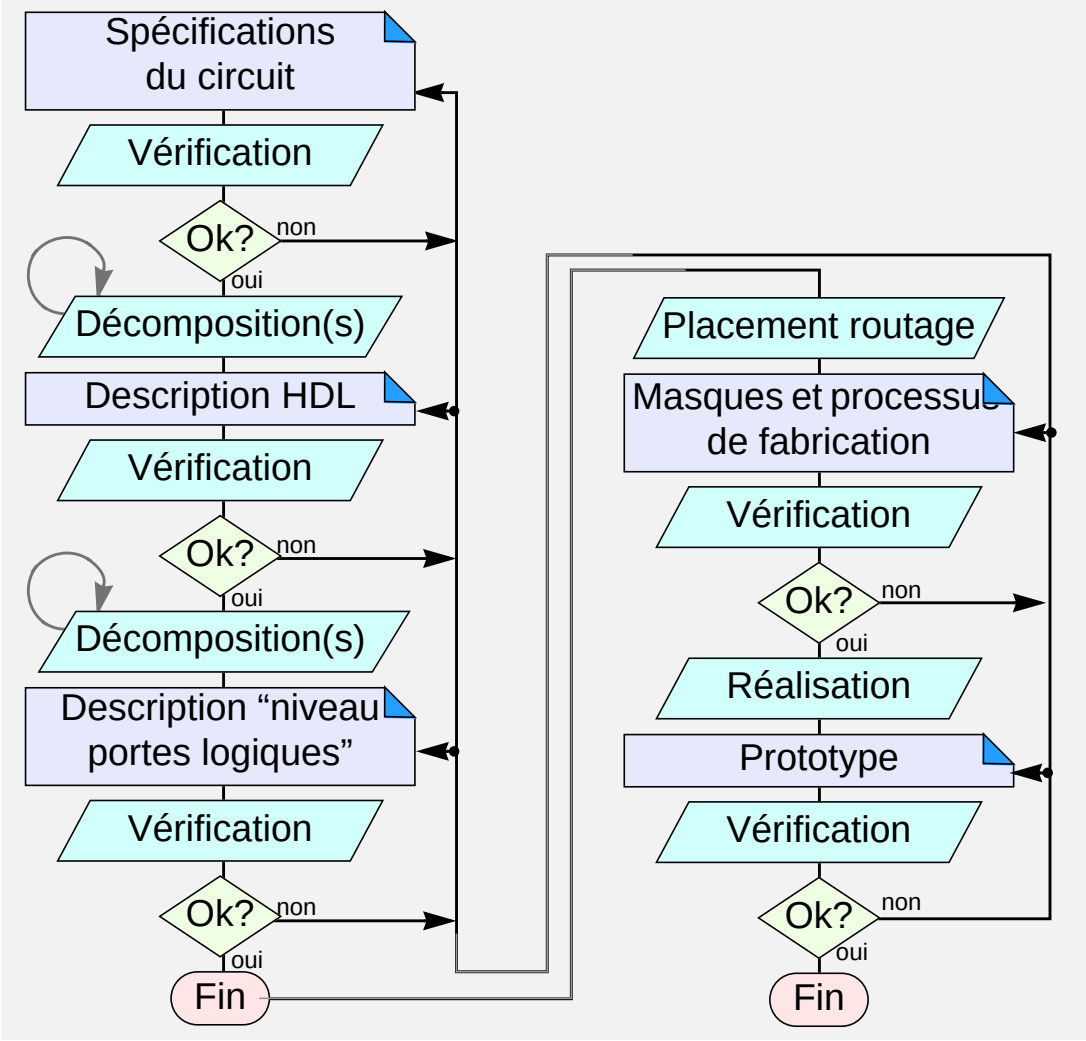


Figure 2-5. Conception descendante d'électronique numérique.

Après vérification de la description HDL du circuit, des outils automatiques permettent d'effectuer les opérations successives de décompositions (généralement appelées *synthèses*) peuvent débuter [Syn99], [Men99], [Cad99]. Le résultat de ces opérations est une description du circuit au niveau des portes logiques (primitives).

Les fabricants de circuits intégrés possèdent des lignes de production basées sur des *technologies* éprouvées. Pour que les outils de décomposition automatique puissent générer la description d'un circuit au niveau des portes logiques d'une certaine technologie, il faut que celles-ci soient définies.

La fin de la conception du circuit est atteinte après vérification du résultat de la décomposition. Cette étape de vérification est réalisée grâce à des outils de simulation. Un modèle comportemental de chaque porte logique de la technologie visée est donc nécessaire à cette étape de vérification.

Avant la fabrication d'un prototype, il reste encore deux étapes à franchir. Il s'agit du placement et du routage des éléments. Pour ces deux étapes, des outils existent et sont couramment utilisés [Men99], [Cad99]. Pour que ces outils fonctionnent, il faut qu'ils soient alimentés par des représentations structurelles de chaque porte logique de la technologie visée. Les plans de masques (*layout*) utiles à la fabrication des portes logiques (nommés *cellules standards*) d'une certaine technologie symbolisent la représentation structurelle de celles-ci. En effet, en partant des plans de masques, et grâce à une description des processus de fabrication, on peut retrouver les dimensions physiques des portes logiques.

Lorsque pour une certaine technologie la représentation structurelle et comportementale des portes logiques existe, nous parlons de *design-kit** (fig. 2-6). Pour créer un design-kit ou plus généralement une bibliothèque de primitives, l'approche ascendante est adoptée. En effet, on part d'éléments de base connus et caractérisés, tels que des transistors et des capacités, pour composer des portes logiques que l'on va caractériser. La composition

revient à construire le plan de masque et la caractérisation à créer le modèle comportemental d'une porte.

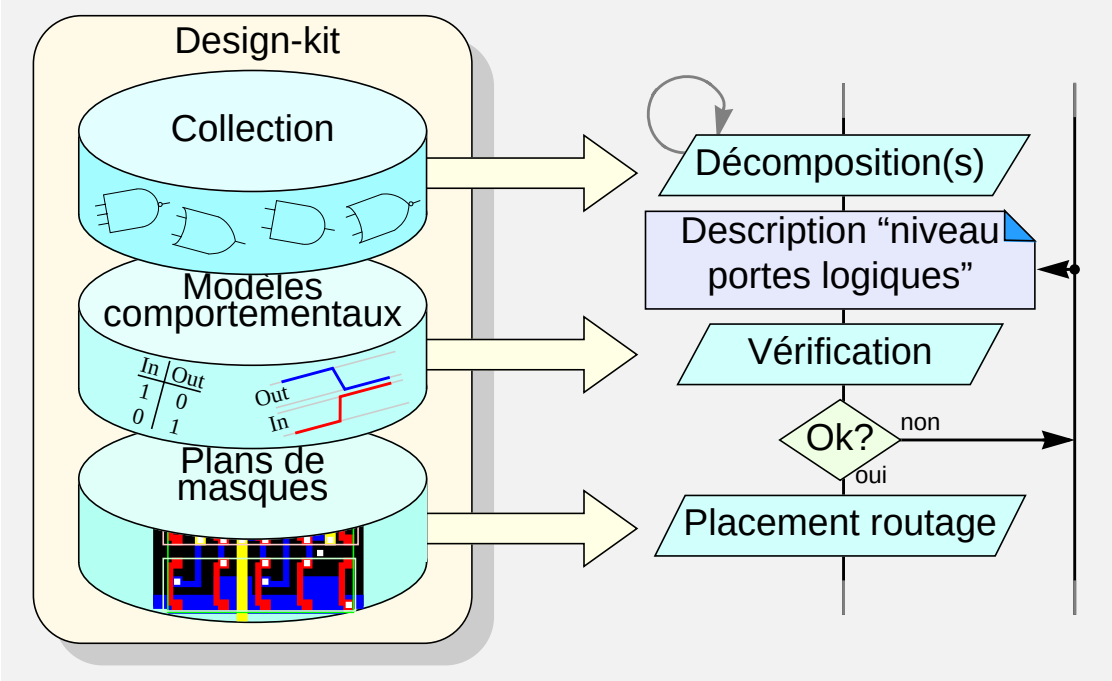


Figure 2-6. Conception descendante et design-kit

Finalement, avant de fabriquer un prototype, il faut encore vérifier le placement et le routage des cellules standards (portes, bascules, etc.). Enfin, lorsque le prototype est réalisé, il faut le tester.

* *La notion de design-kit sera utilisée de façons différentes dans le cas des microsystèmes.*

2.2.3 Conception descendante d'électronique analogique

Pour la conception descendante d'électronique analogique, le point de départ est également un cahier des charges contenant les spécifications du circuit (fig. 2-7). Après vérification des spécifications, les opérations de décompositions peuvent débuter.

Depuis le milieu des années 80, des travaux ont été entrepris pour automatiser la décomposition d'électronique analogique [Gie91], [Van98], [Ant99]. Pourtant en pratique, c'est le plus sou-

vent manuellement que le concepteur réalise les décompositions jusqu'au niveau transistor. La fin de la conception descendante est atteinte lorsque le résultat des décompositions est vérifié.

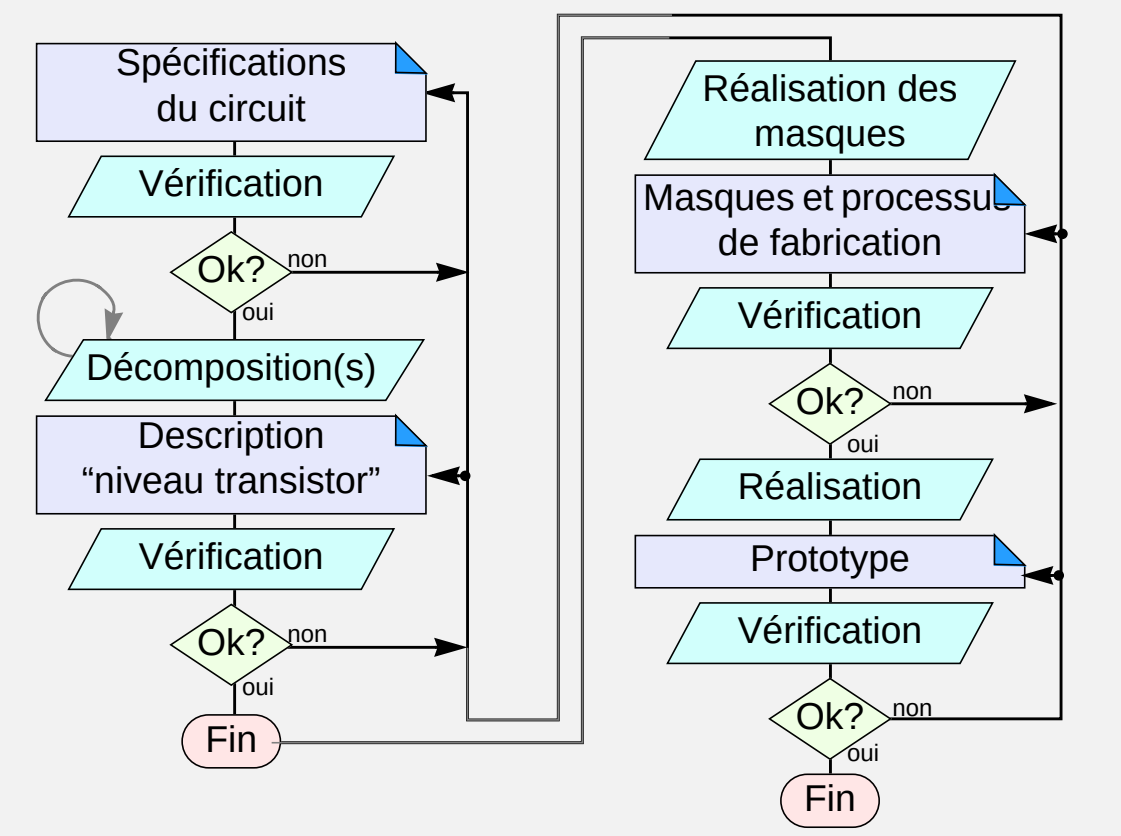


Figure 2-7. Conception descendante d'électronique analogique.

Comme pour la conception descendante d'électronique numérique, avant la fabrication d'un prototype, les plans de masques doivent être conçus et vérifiés. Ici également, des outils existent et permettent d'automatiser ces tâches [Meu96].

2.2.4 Conception de microstructures

Actuellement, comment conçoit-on les microstructures? A notre connaissance, c'est toujours l'approche ascendante qui est adoptée.

Prenons l'exemple d'une microstructure électromécanique (capteur de pression, accéléromètre, gyromètre, etc.). Pour concevoir une telle microstructure, le spécialiste façonne une ébauche de

microstructure basée sur des calculs relativement simples telle que l'estimation des fréquences de résonance, de l'élasticité, des masses, etc. [Sen98a]. Cette opération correspond à l'opération de composition de la conception ascendante (fig. 2-4).

Puis des simulations utilisant la méthode des éléments finis (*EF*) sont effectuées [Aba99] [Ans99] [Mem99]. Le but est de caractériser la microstructure. Si le résultat n'est pas satisfaisant, la conception est affinée.

Ensuite, il faut réaliser les plans de masques ainsi que les processus de fabrication permettant de construire la microstructure. Pour cette étape, des outils existent [Sen92] [Kar96] [Mem99], ils permettent de simuler les processus de fabrication et d'extraire une représentation en trois dimensions de la microstructure. Celle-ci peut être simulée en utilisant la méthode des EF. Finalement, un prototype peut être construit puis caractérisé afin de valider la conception de la microstructure.

Dans le cadre des microsystèmes, et de façon comparable à ce qui est accompli en électronique numérique et analogique, des bibliothèques de microstructures (pouvant être ajustées grâce à des paramètres) peuvent être confectionnées. Celles-ci sont utilisées lors de la conception de microsystèmes. Dans le chapitre 4, nous verrons ce que l'on entend par bibliothèque de microstructure et comment en constituer.

2.2.5 Conception descendante de microsystèmes

Précédemment, nous avons vu comment on conçoit séparément de l'électronique numérique, de l'électronique analogique et des microstructures. Nous avons laissé de côté la discussion des outils et méthodes mis en oeuvre pour réaliser une (ou plusieurs) partie SW. Ce que nous voulons traiter ici, ce sont les problèmes rencontrés lors de la conception descendante de microsystèmes.

Imaginons donc que l'on ait à concevoir un microsystème composé de microstructures, d'électronique analogique et d'électroni-

que numérique (fig. 2-8). Comme il s'agit d'un système relativement complexe, il est raisonnable de suivre l'approche descendante. Après définition et vérification des spécifications, il faut décomposer le microsystème en sous-systèmes. Il s'agit de la première opération de décomposition (cf. § 2.2.1). Cette opération est effectuée manuellement. A la fin de cette opération, nous obtenons des spécifications pour chacun des sous-systèmes.

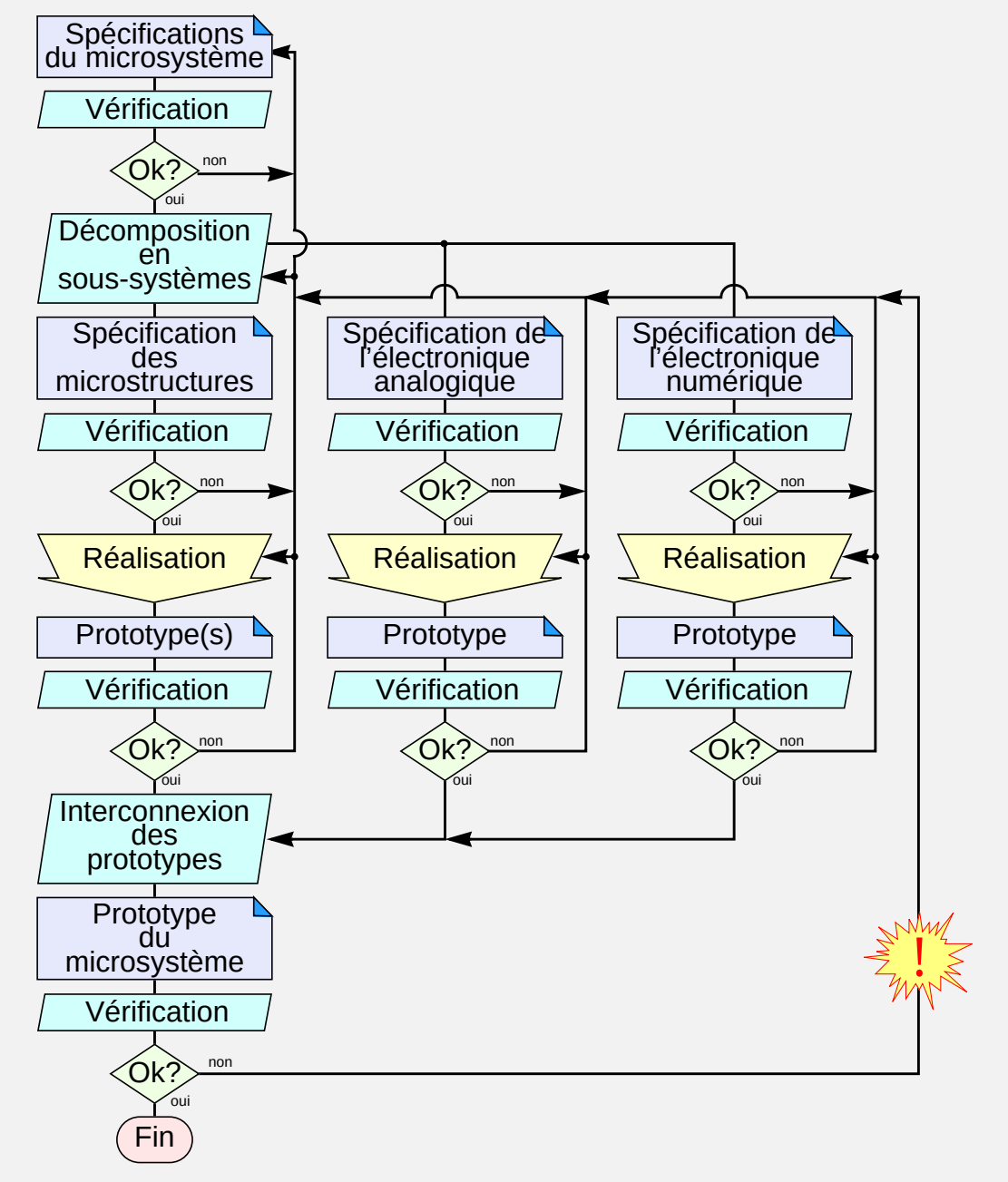


Figure 2-8. Conception descendante de microsystèmes, itérations coûteuses, entreprises lorsqu'un problème de décomposition du microsystème en sous-systèmes est rencontré.

Les spécialistes des différents domaines sont ensuite chargés de concevoir (généralement de façon indépendante) les *sous-systèmes* en respectant les spécifications. Il peut arriver qu'un spécialiste soit chargé de réaliser l'électronique analogique et numérique. Dans ce cas, il est probable que l'interconnexion de l'électronique analogique et numérique ne pose pas de problèmes. En effet, des environnements de développement automatique d'électronique (Electronic Design Automation*: EDA*) existent. Ils permettent de réaliser des simulations multi-modes et, par conséquent, offrent un moyen fiable de contrôler le résultat d'une décomposition.

Il est par contre moins probable qu'une même personne réalise les microstructures ainsi que l'électronique. Si des simulations multi-natures et multi-modes ne sont pas effectuées lors du développement des microstructures et de l'électronique, c'est au moment où les prototypes sont interconnectés que des problèmes seront tardivement constatés. Si les prototypes sont testés individuellement et qu'ils correspondent aux spécifications, cela signifie que s'il y a un problème, celui-ci se situe au niveau de la décomposition du microsystème en sous-systèmes. Il faut alors revoir la décomposition et au mieux recommencer la conception d'un des sous-systèmes.

Le temps et les coûts d'une itération allant des spécifications des sous-systèmes jusqu'à la fabrication de nouveaux prototypes sont importants. Cette approche n'est donc pas satisfaisante!

L'expérience nous montre que les spécialistes des différents domaines ont de la peine à communiquer. A force de travailler dans un domaine, certaines notions deviennent tellement familières qu'elles sont omises dans les spécifications! Pour réaliser un microsystème et afin de minimiser les problèmes rencontrés lorsque les différents prototypes sont mis ensemble, nous proposons d'adopter l'approche MicroSIM.

2.3 Approche *MicroSIM*

Le but de l'approche MicroSIM est de proposer une méthode pour réaliser des microsystèmes, et de promouvoir un environnement CAO pour microsystèmes. La méthode proposée est de type descendante (cf. § 2.2.1). Comme le montre grossièrement la figure 2-9, cette méthode se compose de deux étapes. La première étape est appelée *modélisation comportementale* et la deuxième *modélisation post-design*.

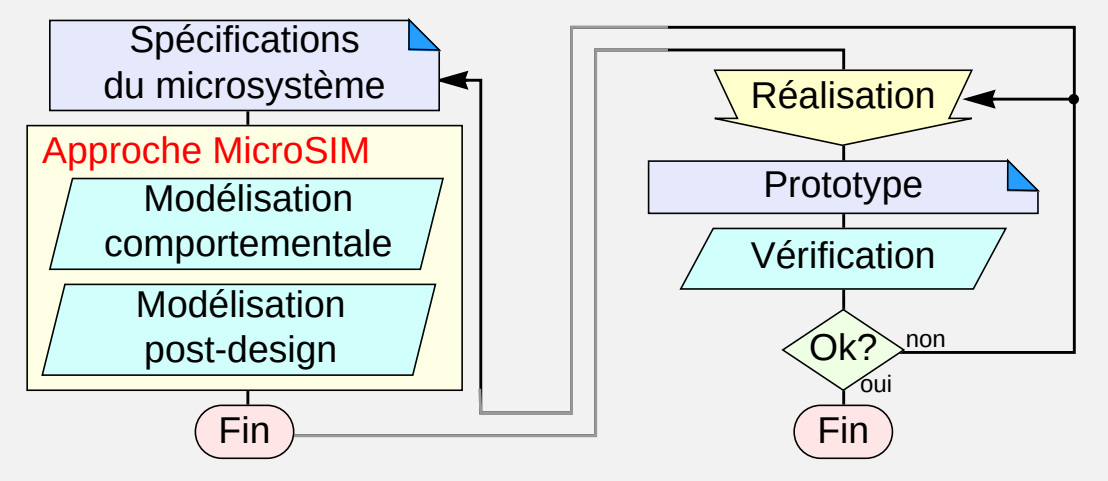


Figure 2-9. Conception descendante, approche MicroSIM.

Dans les paragraphes suivants, nous présenterons ces deux étapes tout en discutant des implications au niveau de l'environnement CAO. Nous passerons en revue les simulateurs et les modèles dont nous avons besoin durant la conception d'un microsystème.

2.3.1 Modélisation comportementale

Imaginons que l'on ait un microsystème à concevoir. Dans l'approche MicroSim, la première étape est appelée modélisation comportementale. L'idée est d'utiliser un environnement CAO afin d'effectuer le plus rapidement possible des simulations du microsystème. Le but de cette étape n'est pas d'utiliser ou de développer des modèles bas-niveau des différentes parties du microsystème. Les buts de cette étape sont:

1. Effectuer une décomposition en sous-systèmes de l'architecture en utilisant des modèles comportementaux haut-niveau des sous-systèmes.
2. Valider la décomposition en sous-systèmes grâce à la simulation.
3. Obtenir des modèles comportementaux des sous-systèmes pouvant être utilisés comme spécification lors de la deuxième étape de l'approche MicroSIM (modélisation post-design).
4. Développer des bancs de tests utilisables durant toute la deuxième étape de l'approche MicroSIM (voire même lors de la vérification du prototype).

Les modèles des bancs de tests doivent permettre de garantir les spécifications du microsystème tout au long de son développement. Les simulations globales peuvent servir à démontrer la faisabilité du microsystème à des non-spécialistes de la conception de microsystèmes: décideurs, spécialistes de la commercialisation, experts d'autres disciplines, etc.

Cette première étape (fig. 2-10) se compose d'une opération de vérification des spécifications et d'une décomposition. La décomposition a pour but de diviser le système en sous-systèmes et de développer des bancs de tests. Les résultats de la décomposition sont les spécifications des sous-systèmes et des bancs de tests. A chaque sous-système et chaque banc de tests, un modèle comportemental est attribué. Ensuite des simulations sont effectuées de manière à vérifier les spécifications du microsystème. Pour cette première étape, l'environnement CAO doit permettre d'effectuer des simulations globales multi-natures et multi-modes de modèles haut-niveau.

Une bibliothèque de modèles comportementaux de sous-systèmes (contenant éventuellement leurs symboles) constitue le *design-kit* pour microsystèmes de cette étape de modélisation comportementale. La bibliothèque de sous-systèmes doit contenir des modèles de microstructures ainsi que des modèles d'algorithmes pour le traitement du signal. Chaque modèle comportemental doit posséder des paramètres pouvant être ajustés afin de correspondre

au problème rencontré. Durant la phase de modélisation comportementale, les paramètres permettent également d'expérimenter différentes solutions. Lorsqu'une solution est choisie, celle-ci est prise comme point de départ pour la prochaine phase.

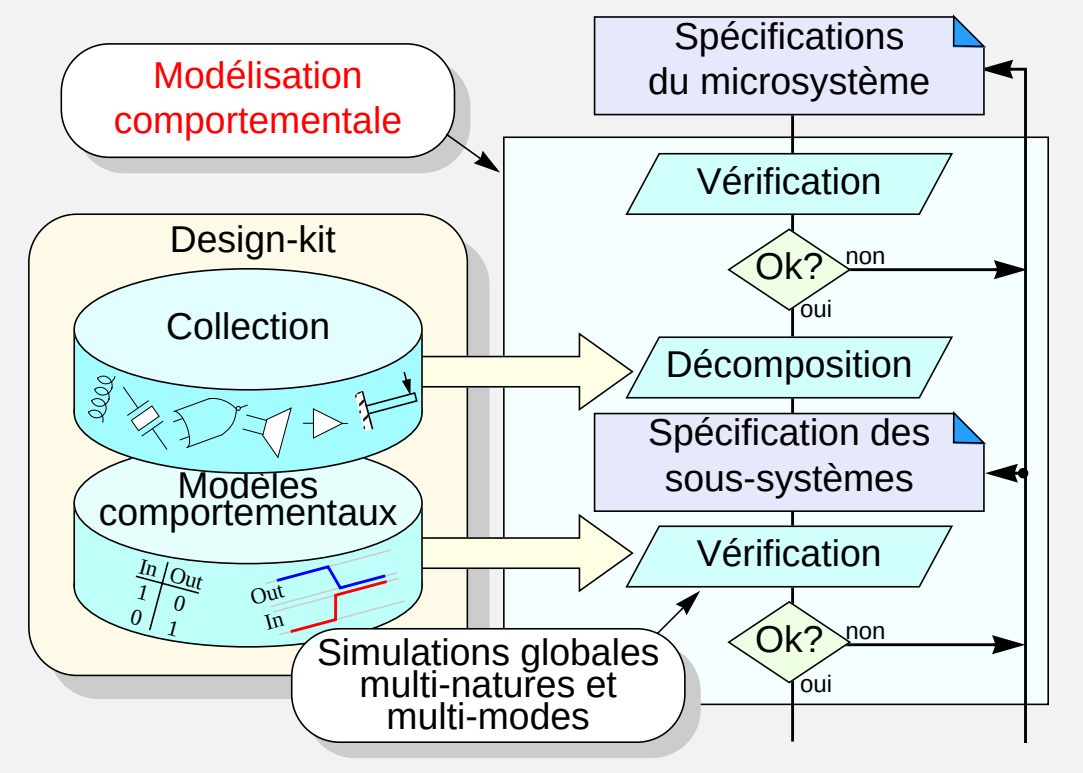


Figure 2-10. Approche MicroSIM, modélisation comportementale d'un microsystème.

Dans le chapitre 4, nous présenterons des exemples de modèles ainsi que des méthodes et des outils permettant d'obtenir des modèles comportementaux.

2.3.2 Modélisation post-design

Lorsque l'architecture du microsystème a été simulée avec succès, les modèles comportementaux des sous-systèmes servent de spécification lors de la deuxième étape de modélisation. Cette seconde étape de modélisation est nommée post-design, dans la mesure où la conception de l'architecture du microsystème est figée.

Comme pour l'étape précédente, ce sont les méthodes de conception descendante qui sont utilisées dans les différents domaines touchés (cf. § 2.2.2, § 2.2.3 & § 2.2.4). Ainsi, des opérations de décompositions successives ont lieu. Après chaque décomposition une simulation du microsystème entier est effectuée. Celle-ci permet de vérifier le résultat de la décomposition globale.

Lorsque tous les sous-blocs ont atteint une description structurelle, cette seconde phase est terminée. La réalisation de prototypes peut commencer.

Durant la modélisation post-design, l'environnement CAO doit permettre de réaliser des simulations globales, multi-natures, multi-modes et multi-niveaux. Dans le cas où un logiciel doit être développé en même temps qu'une partie HW (par exemple un microcontrôleur), l'environnement de simulation doit également permettre d'effectuer des co-simulations HW-SW.

2.4 Conclusion

Les microsystèmes sont composés de sous-systèmes complexes, hétérogènes et possédants souvent de forts couplages entre eux. Pour réaliser un microsystème, la méthode qui consiste à le diviser en sous-systèmes puis à développer les sous-systèmes séparément pour finalement les réunir n'est pas satisfaisante. Le risque qu'une itération soit nécessaire est trop élevé.

Notre réponse à ce problème est l'approche MicroSIM. L'approche MicroSIM est une méthode basée sur des outils de CAO. Pour appliquer cette méthode, nous avons vu que les outils de simulation de l'environnement de CAO doivent admettre des simulations globales, multi-natures, multi-modes, multi-niveaux ainsi que la co-simulation HW-SW. Les contraintes au niveau des simulateurs sont donc importantes. Le principal avantage de cette méthode est d'offrir un grand contrôle durant les différentes phases de la conception, allant des spécifications jusqu'au prototypage. Elle permet également de démontrer la faisabilité d'un microsystème très tôt dans le processus de développement. La modélisation com-

portementale incite les spécialistes des différents domaines à communiquer entre eux. Dans cette méthode, il est important de développer rapidement des bancs de tests. Ceux-ci servent tout au long de la conception du microsystème.

Cette méthode comporte également des désavantages. En effet, pour qu'elle soit applicable, il faut un environnement CAO mais également un design-kit. Le design-kit doit posséder de bons modèles comportementaux. Or, l'élaboration de bons modèles comportementaux de microstructures n'est pas une opération triviale. Dans le chapitre suivant, nous discuterons de certains problèmes rencontrés lors de l'élaboration de ce genre de modèles.

Une limitation de cette approche est le temps de simulation. Lorsque les modèles des sous-systèmes deviennent détaillés (modèles bas-niveau), les temps de simulation peuvent devenir extrêmement longs. Il faut alors trouver le bon compromis entre le niveau de détail et le temps de simulation.

Parfois également, un microsystème peut posséder des constantes de temps très différentes. C'est le cas lorsqu'une partie d'un sous-système travaille vite alors qu'une autre partie travaille lentement. Dans un tel cas les temps de simulations explosent et cette approche devient plus difficile à appliquer.

2.5 Références

[Aba99] *"ABAQUS, Products at Hibbitt, Karlsson & Sorensen, Inc."*, http://www.hks.com/

[Ama99] H.-P. Amann, A. Boegli, V. Moser, C. Berthoud, M. Ansorge, F. Pellandini, *"Modelling and Simulation for Sensor Interfaces : the example of a "Smart" Pressure Sensor"*, Sensorama 99, Yverdon, Switzerland, Oct. 27-28, 1999

[Ans99] *"ANSYS, Products at CADfix"*, http://www.ansys.com/

[Ant99] *"Antrim Design Systems, Inc. provides tools and methods for the development of Mixed-Signal Integrated Circuits"*, http://www.antrim.com

[Boe96a] A. Boegli, V. Moser, H.-P. Amann, F. Pellandini, B. Romanowicz, Ph. Lerch, M. Laudon, Ph. Renaud, *"System-Level Simulation Using Mixed-Nature Models"*, 1st Europe-Asia Congress on Mechatronics, Besançon, France, Oct. 1-3, 1996, p.12-1

[Cad99] *"Technology Solutions at CADENCE"*, http://www.cadence.com/

[Gie91] G.Gielen, W.Sansen, *"Symbolic Analysis for Automated Design of Analog Integrated Circuits"*, Kluwer Academic Publisher, Boston, 1991

[Kar96] J.M.Karam, B.Courtois, K.Hofmann, A.Poppe, M.Rencz, M.Glesner, V.Székely, *"CAD of MEMS: from the idea to the reality"*, Proceedings of the 1st Europe-Asia Congress on Mechatronics, Besançon, France, Oct.1-3, 1996, p.70-75

[Mem99] *"MEMCAD, Products at Microcosm Technologies"*, http://www.memcad.com/

[Men99] *"Products at Mentor Graphics"*, http://www.mentorg.com/

[Meu96] W.Meusburger, H.Senn, P.Söser, *"Development of Device Generators for the support of the Analogue IC Design with Mentor V8"*, 1st ALPS LUG conference, Neuchâtel, Switzerland, Sep. 26-27, 1996

[Mos96] Vincent Moser, *"Computer-Aided Behavioural Modelling of Analogue Systems"*, thèse de doctorat, Université de Neuchâtel, Switzerland, 1996

[Pop95] R.S.Popovic, *"Sensor Microsystems"*, 20th International Conference on Microelectronics, Vol.2, Nis, Serbia, Sep.12-14, 1995, p.531-537

[Sen92] S.D.Senturia, R.M.Harris, B.P.Johnson, S.Kim,K.Nabors, M.A.Shulman, J.K.White, *"A Computer-Aided Design System for Microelectromechanical Systems (MEMCAD)"*, Journal of Microelectromechanical Systems, Vol.1 Nr.1, 1992, p.3-13

[Sen98a] S.D.Senturia, *"Design of Microelectromechanical Devices, A Short Course"*, cours organisé par la FSRM et donné par le professeur S. D. Senturia à Neuchâtel en juin 1998

[Syn99] *"Products & Solutions at SYNOPSYS"*, http://synopsys.com/

[Van98] J.Vandenbussche, S.Donnay, F.Leyn, G.Gielen, W.Sansen, *"Hierarchical top-down design of analog sensor interfaces: form system-level specifications downs to silicon"*, DATE'98, Paris, France, Feb. 23-26, 1998

Chapitre 3

Langages et simulateurs

Le chapitre précédent définit ce que sont la conception descendante et ascendante de systèmes ou sous-systèmes. Ainsi, nous avons vu que la conception descendante est composée de cycles de décomposition-vérification alors que la conception ascendante est formée de cycles de composition-caractérisation. Nous avons également vu que les microsystèmes sont formés de sous-systèmes pouvant être classés dans des groupes (fig. 2-2). Pour chacun de ces groupes, des environnements de CAO ont été développés. Ceux-ci possèdent des simulateurs et des langages de modélisation permettant d'effectuer les vérifications ou caractérisations des cycles de conception.

Notons qu'il existe des différences importantes entre simulateurs des différents groupes. A quoi sont dues ces différences? La réponse à cette question est à chercher dans la physique et les mathématiques utilisées pour modéliser les sous-systèmes.

Dans ce chapitre, nous présenterons les principaux langages de modélisation et simulateurs existants. Cela nous permettra d'en souligner les différences (types d'analyses, types de signaux, etc.) pour finalement être en mesure de proposer une solution compatible avec l'approche MicroSIM.

3.1 Electronique numérique

En électronique numérique la notion d'*évènement* est importante. Il s'agit de la transition d'un état stable vers un autre état stable. Les signaux de ce type de circuits prennent des valeurs quantifiées à des moments donnés (*échantillons*), c'est-à-dire après un évènement. Ce type de signaux présente donc des discontinuités.

Il est possible de représenter la fonctionnalité de ces systèmes grâce à des équations Booléennes, à des machines d'états (*FSM*: Finite State Machine) ou à des réseaux de Petri (Petri net).

Dans ce travail, pour analyser les fonctions de transfert de systèmes numériques, nous utilisons la *transformation en Z*. Celle-ci est notée et définie de la manière suivante:

$$\{x_n\} \square\!\!-\!\!\blacksquare\ X(z) = \sum_{n=0}^{\infty} x_n \cdot z^{-n} \qquad (3.1)$$

où $\{x_n\}$ est une série d'échantillons, X(z) la transformation en Z de la série d'échantillons et z l'*opérateur de la transformation en* Z.

Pour simuler des circuits numériques, il existe toute une gamme de simulateurs. Il s'agit de simulateurs commandés par des évènements (*event-driven simulator*). Ceux-ci permettent d'effectuer des simulations temporelles (*time-domain simulation*), c'est-à-dire transitoires.

Pour modéliser les circuits numériques, deux HDL sont communément utilisés. Il s'agit des langages VHDL et Verilog HDL. Ces deux langages sont des normes, et une abondante littérature leur est consacrée.

VHDL fut développé au début des années 1980. Il découle d'un projet de recherche sur les circuits intégrés très rapides, financé par le département de la défense US. C'est en 1987 que VHDL a été accepté comme norme: IEEE 1076-1987 (*IEEE:* Institute of Electrical and Electronics Engineers, [IEE99a]). Une version améliorée et modernisée de VHDL est sortie en 1994: IEEE 1076-1993

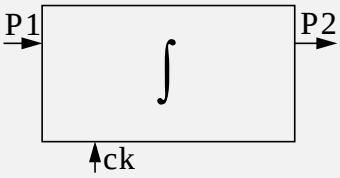
[IEE93]. Verilog HDL est un langage introduit par Gateway Design Automation au milieu des années 1980. C'est en 1995 que Verilog HDL a été accepté comme norme: IEEE 1364-1995 [IEE95]. Ce langage est très usuel aux USA alors qu'en Europe, c'est VHDL qui est le plus souvent utilisé. Généralement, les simulateurs de HDL numériques acceptent ces deux normes.

Ces deux HDL sont utilisés dans le cadre de la conception descendante d'électronique numérique. Ils permettent de modéliser des systèmes numériques avec plus ou moins de détails.

(a) VHDL

```
ENTITY integrator IS
 GENERIC ( Kc : ... );
 PORT    ( P1 : IN  ...
           P2 : OUT ...
           ck : IN  ... );
END ENTITY integrator;
```

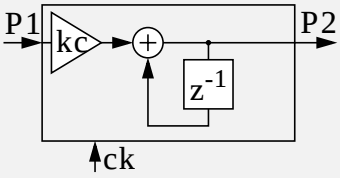
Représentations symboliques



(b) VHDL

```
ARCHITECTURE model1 OF integrator IS
BEGIN
...
END ARCHITECTURE model1;
```

Représentations symboliques




```
ARCHITECTURE model2 OF integrator IS
BEGIN
...
END ARCHITECTURE model2;
```

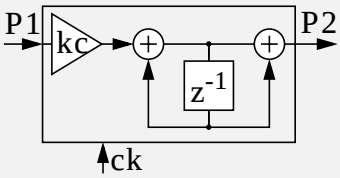


Figure 3-1. (a) Vues externes et (b) internes de modèles, représentations VHDL et symbolique.

A la manière de symboles représentant des schémas, tout modèle VHDL possède deux vues (fig. 3-1):

- une externe, qui montre ses connexions avec le monde extérieur;

- une interne, qui décrit sa réalisation sous forme d'interconnexions d'autres modèles plus simples ou sous forme d'algorithmes.

Un modèle VHDL est une entité, sa vue externe est appelée spécification d'entité et sa vue interne se nomme architecture. Une vue externe peut avoir plusieurs vues internes associées. Ces vues internes (fig. 3-1 b) dépendent souvent du degré de finesse souhaité pour décrire le fonctionnement du modèle. Dans le cadre de l'approche MicroSIM, c'est cette dernière particularité qui est exploitée.

Dans la pratique, il est possible de créer des modèles ne pouvant être synthétisés (décomposés) automatiquement. C'est-à-dire que les simulations d'un modèle peuvent correspondre à un cahier des charges sans que les outils de synthèse automatique ne trouvent de solutions sous la forme de schémas de portes logiques (primitives). Pour développer un modèle pouvant être synthétisé automatiquement avec un bon résultat, il est nécessaire d'utiliser un certain style de programmation basé sur un vaste savoir-faire.

3.2 Electronique analogique

Le monde de l'électronique analogique est très différent de celui de l'électronique numérique. En électronique numérique le temps est rythmé par les évènements et les valeurs sont quantifiées. En électronique analogique les courants et tensions sont continus par rapport au temps (définition de la continuité d'une fonction, voir [Pis80]). Des systèmes d'équations différentielles ordinaires (*ODE:* Ordinary Differential Equation) et algébriques (*ODAE*: Ordinary Differential and Algebraic Equation) permettent de décrire le comportement de ce type de circuits. On dit que f(x)=0 est une *équation algébrique* si f(x) est un polynôme.

Une ODE est une équation impliquant une fonction et ses dérivées:

$$F(x;y;y';\ldots;y^{(n)}) = 0 \quad \& \quad y^{(n)} = \frac{d^n y}{dx^n} \qquad (3.2)$$

En électronique analogique, x correspond au temps.

Dans ce travail, pour analyser ce type de circuits, nous utilisons la *transformation de Laplace*. Celle-ci est notée et définie de la manière suivante:

$$x(t) \circ\!\!-\!\!\bullet\ X(s) = \int_{+0}^{\infty} x(t) \cdot e^{-st} \cdot dt \qquad (3.3)$$

où x(t) est une fonction définie sur l'axe temporel positif, X(s) la transformation de Laplace de x(t) et s l'*opérateur de Laplace*.

Pour modéliser les circuits électroniques analogiques, on utilise généralement des réseaux de Kirchhoff [Boi83]. Les réseaux de Kirchhoff sont constitués par la connexion d'un nombre fini d'éléments. Les éléments possèdent des bornes destinées à établir la connexion. Chaque borne est caractérisée par deux grandeurs: le potentiel (*across*) et le courant (*through*) [Boi83]. Etablir la connexion de plusieurs bornes revient à faire coïncider le potentiel de ces bornes. Les réseaux de Kirchhoff sont des systèmes dits *conservatifs*. C'est-à-dire que la somme des puissances absorbées par toutes les branches d'un réseau de Kirchhoff est identiquement nulle, quelle que soit la nature (linéaire, non-linéaire, autonome, non-autonome) des éléments (primitives) constituant le réseau et quelle que soit leur interconnexion.

Dans le cas des circuits électroniques analogiques, la grandeur "potentiel" est une tension électrique exprimée en volt, alors que la grandeur "courant" est un courant électrique exprimé en ampère. Les bornes connectées ensemble sont appelées noeuds du circuit. Ainsi le modèle d'un circuit électronique analogique est composé d'une liste d'éléments, d'une liste de noeuds et d'une description comportementale de chacun des éléments. Cette modélisation contient le système d'ODAE exprimant la loi de conservation de l'énergie: loi des courants de Kirchhoff et loi des tensions de Kirchhoff.

Pour simuler ce genre de modèles on utilise des simulateurs de temps continus de type *SPICE* (Simulation Program With Integrated Circuit Emphasis [Nag75]). Ceux-ci exécutent différents types de simulations. Les trois principaux sont:

- la simulation en régime continu (*DC*: Direct Current) qui permet de déterminer un point de fonctionnement;
- la simulation en régime alternatif (*AC:* Alternating Current) également appelée analyse petits signaux qui permet d'obtenir la réponse d'un système lorsqu'un signal sinusoïdal est appliqué à l'entrée;
- l'analyse transitoire qui donne la réponse temporelle d'un système à un signal d'entrée arbitraire.

Les simulateurs de type SPICE possèdent une bibliothèque d'éléments pouvant être paramétrés, tels que résistances, capacités, transistors, etc. Celle-ci est utilisée pour créer et simuler des modèles de circuits électroniques analogiques.

3.2.1 Langage SPICE

La figure 3-2 nous montre un modèle d'une source de courant sous une forme symbolique (schéma) et sous la forme d'un fichier SPICE (*netlist*). Le fichier *SPICE* contient la liste des éléments suivis des noeuds et des paramètres des éléments. Le paramètre de la résistance est sa valeur en ohm. Pour les transistors, les paramètres sont sa largeur et sa longueur.

Une évolution de ce langage permet de créer des sous-circuits (*subckt*: sub-circuit). Cela permet de créer des modèles hiérarchisés de circuits analogiques. Le langage SPICE est conçu pour être utilisé dans le cadre de l'approche ascendante.

Généralement, les simulateurs de type SPICE ne permettent pas d'étendre les bibliothèques d'éléments. C'est sans doute parce que ces simulateurs sont consacrés à l'électronique analogique et que les primitives de l'électronique analogique sont relativement faciles à identifier. En outre, le but de ces simulateurs est de trai-

ter le plus d'éléments possibles. Les modèles d'éléments ont donc été conçus afin de minimiser les temps de simulations. Tâche qui n'est pas triviale et qui est généralement réservée à des spécialistes [Fot97].

(a)

```
V_vcc vcc 0 DC 2.6
V_vss vss 0 DC 0
R vr vss 85k
m4 vcc vpp vm vcc P w=10u l=5u
m5 vcc vpp vpn vcc P w=5u l=5u
m3 vcc vpp vpp vcc P w=5u l=5u
m6 vpn vpn vss vss N w=5u l=5u
m2 vm vm vss vss N w=20u l=2u
m1 vpp vm vr vss N w=20u l=2u
.end
```

(b)

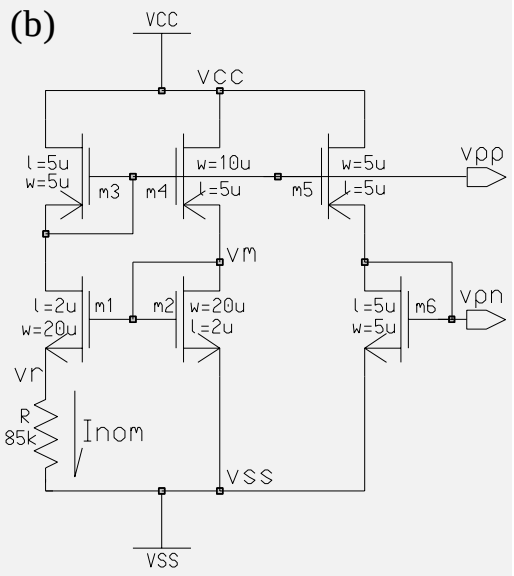


Figure 3-2. Modèle SPICE d'une source de courant, (a) fichier SPICE (netlist) et (b) schéma.

Les modèles construits en assemblant des éléments idéaux tels que résistances, conductances, capacités, inductances et sources constantes ou commandées, sont appelés des *macro-modèles*.

La figure 3-3 nous montre le symbole et le schéma d'un macro-modèle d'un transistor MOSFET [Deg94]. Ce macro-modèle est composé de trois éléments : une conductance (dont la valeur est g_0) et deux sources de courant commandées par des tensions (V_G et V_S). Comme nous le verrons plus loin cette méthode de modélisation n'est pas restreinte au domaine de l'électronique analogique.

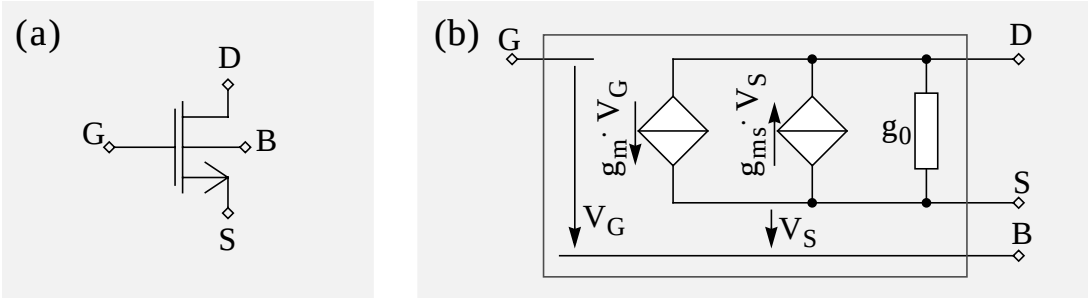


Figure 3-3. (a) Symbole et (b) macro-modèle d'un MOSFET de type N.

Nous l'avons mentionné précédemment: les courants et tensions de circuits électroniques analogiques sont des fonctions temporelles continues. Or, un système d'ODAE permet de modéliser des systèmes continus. A l'origine, les simulateurs de types SPICE ont été conçus pour simuler des modèles strictement continus.

Les simulations d'électronique numérique effectuées par des simulateurs commandés par des évènements, présentent des discontinuités. Ces discontinuités proviennent de simplifications apportées aux modèles des portes logiques numériques. Le but d'une simulation numérique n'est pas de déterminer la forme des transitions d'un état à un autre, mais le moment d'une transition et l'état atteint. Parfois, les transitions sont considérées comme instantanées. Cette constatation révèle quelque chose de fondamental en modélisation: la simplification d'un comportement peut introduire des discontinuités au niveau de la fonction modélisée ou de ses dérivées. Or, une discontinuité ne peut pas être modélisée avec un seul système d'ODEA. Pour permettre la conception de systèmes présentant des discontinuités, des modèles ont été ajoutés aux bibliothèques SPICE. Cette évolution n'est pas sans conséquences sur les simulateurs de type SPICE. Les noyaux de ces simulateurs ont dû être modifiés afin d'accepter la simulation de discontinuités.

3.2.2 Langage HDL-A

Depuis quelques années, certains simulateurs de type SPICE permettent l'extension de la bibliothèque d'éléments. Pour coder de nouveaux éléments, des langages ont été développés. Le langage HDL-A de ANACAD [ANA94] en est un exemple. HDL-A est un HDL permettant de créer des éléments analogique et numérique. Le simulateur supportant ce langage permet donc la réalisation de simulations multi-modes.

A la manière de VHDL, les modèles ou éléments HDL-A possèdent deux vues: une externe et une interne (cf. fig. 3-4 et annexes C). A la différence de VHDL, la vue interne des modèles HDL-A peut contenir deux blocs: un bloc contenant la description de la

partie analogique du modèle (**RELATION**), c'est-à-dire les systèmes d'*ODEA*, et un bloc de processus (**PROCESS**) décrivant le comportement de la partie numérique.

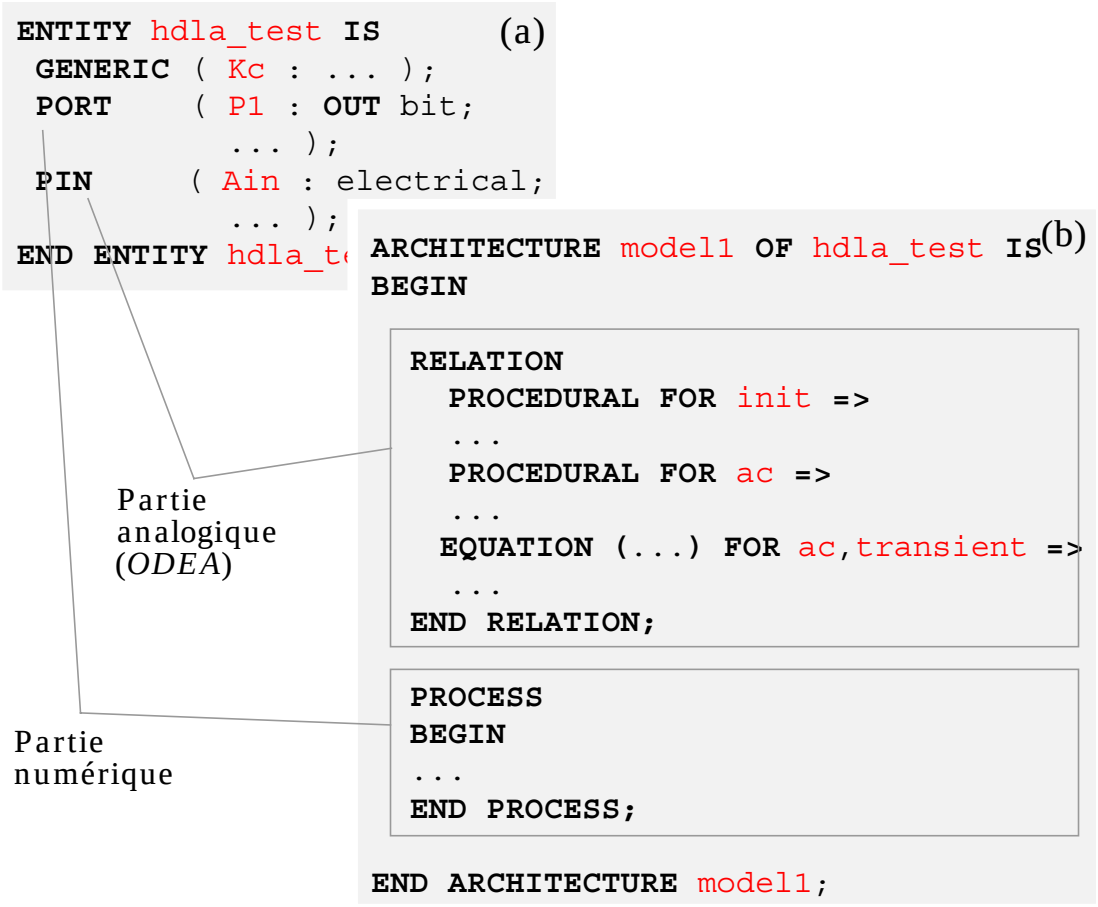


Figure 3-4. (a) Vue externe et (b) interne d'un modèle HDL-A.

La partie numérique de *HDL-A* n'est pas aussi développée que VHDL. La partie analogique est, elle-même, divisée en blocs de procédures (**PROCEDURAL**) formant autant de vues de cette partie. Chacun des blocs de procédures décrit une partie du comportement du modèle. Le bloc "**INIT**" est utilisé lors de l'initialisation, le bloc "AC" lors de simulations AC, etc. Pour que des simulations de types AC, DC et transitoires puissent être entreprises, il faut que les blocs correspondants soient définis. Une partie numérique ne peut subir de simulations DC ou AC. Lors de simulations de ce type, la partie numérique est inerte. En plus des blocs de procédures, il est possible d'ajouter un bloc contenant des ODEA. Il s'agit du bloc nommé **RELATION**. Bien entendu, il faut que les différentes vues du modèle soient cohérentes entre elles.

3.2.3 Langage VHDL-AMS

Les langages tels que HDL-A ou MAST de Analogy sont des langages appartenant à des entreprises. Depuis quelques années, des efforts ont été consentis afin de normaliser un HDL analogique ou mixte (analogique-numérique). Ces efforts ont récemment (1999) abouti à l'extension de la norme IEEE 1076-1993 nommée IEEE 1076.1-1999 ou *VHDL-AMS* (VHDL with Analog and Mixed-Signal extensions) [IEE99b]. Notons que des efforts similaires sont actuellement entrepris afin de produire une norme IEEE pour Verilog-AMS.

Les premiers compilateurs et simulateurs VHDL-AMS viennent de sortir [Ana99][Cin99][Men99], c'est pour cette raison que la majorité des modèles que nous présentons dans ce travail n'est pas codée avec ce nouveau langage.

VHDL-AMS possède une sémantique permettant la description et la simulation de systèmes électroniques ou non-électroniques numériques, analogiques ou mixtes (analogiques-numériques). La partie numérique de VHDL-AMS contient entièrement la norme IEEE 1076-1993. La partie analogique permet de modéliser des systèmes à constantes localisées (*lumped systems*) à différents niveaux d'abstraction. Un système à *constantes localisées* est un système dont les propriétés physiques sont localisées en un point. Le comportement de ce type de systèmes peut être décrit par des systèmes d'ODEA. Inversement un système à *constantes réparties* est un système dont les propriétés physiques sont réparties dans l'espace. Les systèmes à constantes réparties sont décrits par des équations différentielles partielles (*PDE:* Partial Differential Equation, voir § 3.4)

Le langage VHDL-AMS permet donc la simulation globale, multi-niveaux, multi-natures et multi-modes dont nous avons besoin pour appliquer l'approche MicroSIM.

La figure 3-5 (voir également les annexes C) nous montre deux vues partielles (externe et interne) d'un modèle électronique mixte (analogique-numérique) codé avec VHDL-AMS. Contrairement à

HDL-A, toutes les entrées/sorties sont des **PORT**s. En VHDL-AMS, il existe trois types de **PORT**s:

- **SIGNAL**, pour les entrées/sorties concernant le simulateur d'évènements (numérique).
- **TERMINAL**, pour les entrées/sorties analogiques respectant les lois des réseaux de Kirchhoff (principe de conservation de l'énergie respecté au niveau des noeuds).
- **QUANTITY**, pour les entrées/sorties analogiques caractérisées par une seule grandeur. Ces entrées/sorties permettent une représentation en diagramme de bloc (*block diagram*), représentation non-conservative au niveau des noeuds (fig. 3-6).

(a)

```
ENTITY vhdlams_test IS
 GENERIC ( Rs : Real;
           ... );
 PORT    ( SIGNAL    P1 : IN  bit;
           THERMINAL P2 : electrical;
           QUANTITY  P3 : OUT real;
           ... );
END ENTITY vhd
```

(b)

```
ARCHITECTURE model1 OF vhdlams_test IS
  quantity VP2 across IP2 through P2;
  ...
BEGIN
    VP2 == IP2 * Rs;  -- loi d'Ohm
    P3  == VP2 * IP3; -- Puissance dissipée
    ...
pr1: PROCESS(P1)
     BEGIN
     ...
     END PROCESS pr1;
    ...
END ARCHITECTURE model1;
```

Figure 3-5. (a) Vue externe et (b) interne d'un modèle VHDL-AMS.

HDL-A a été conçu pour permettre de créer des modèles utilisables dans des fichiers de type SPICE. VHDL-AMS ne s'oppose pas à cette approche. Certains simulateurs [Men99] permettent de mélanger des modèles VHDL-AMS avec des éléments d'une biblio-

thèque SPICE de façon hiérarchisée. Au haut de la hiérarchie on peut trouver soit un modèle VHDL-AMS, soit un fichier de type SPICE.

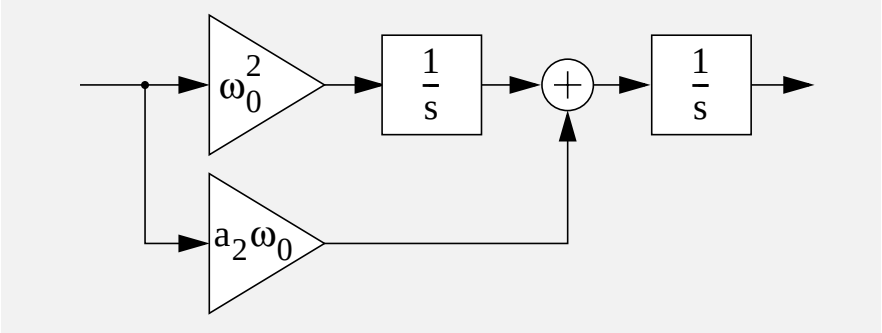


Figure 3-6. Exemple d'un diagramme bloc. Il s'agit d'un filtre de boucle analogique du deuxième ordre [Kap96].

3.3 Electronique mixte

Nous venons de le voir, les simulateurs de l'électronique numérique (simulateurs commandés par des évènements) et les simulateurs analogiques (simulateurs de temps continus) sont différents et n'ont en commun qu'un type de simulation: l'analyse transitoire.

La figure 3-7 nous montre le schéma d'un circuit composé d'éléments d'électronique numérique et analogique. La question que nous pouvons nous poser est la suivante: pourquoi a-t-on besoin de simulateurs si différents alors qu'il n'y a pas de distinction fondamentale entre électroniques analogique et numérique? En effet, il est possible de constater que tous les éléments électroniques (analogiques et numériques) respectent les mêmes lois physiques.

La réponse à cette question est à chercher dans les performances des deux types de simulateurs. Si les simulateurs d'électronique analogique étaient plus rapides, les simulateurs d'électronique numérique ne seraient pas nécessaires. Malheureusement ce n'est pas le cas et les parties numériques sont modélisées en utilisant des modèles approximatifs. Ceux-ci sont approximatifs dans le sens où ils ne précisent pas les caractéristiques opérationnelles. Les modèles numériques sont représentés

par des descriptions d'états de type booléen, sans ressembler au circuit analogique qui le constitue. Grâce à cette approche, les simulations numériques peuvent être des milliers de fois plus rapides que les simulations analogiques équivalentes, tout en conservant un niveau de précision suffisant pour contrôler une fonctionnalité. Ainsi pour développer une électronique numérique, nous utilisons le langage VHDL et un simulateur commandé par des évènements.

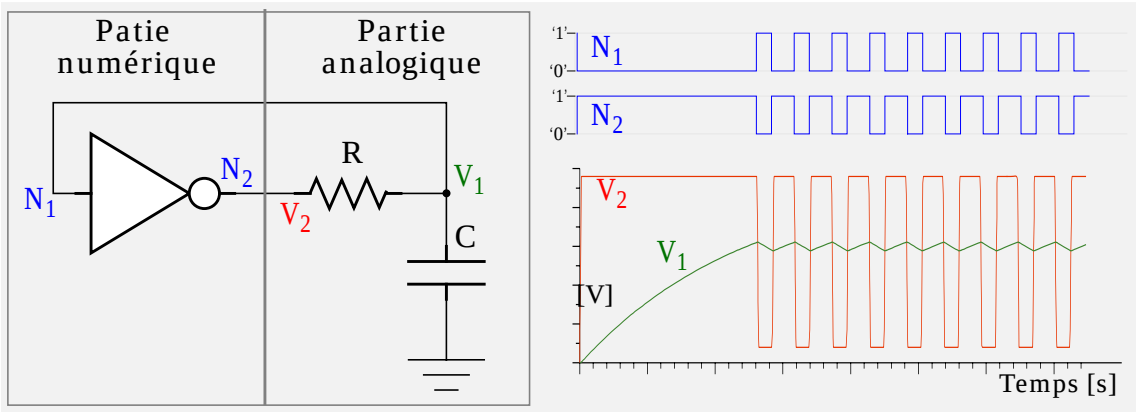


Figure 3-7. A gauche: schéma d'un circuit électronique numérique et analogique. A droite: résultats d'une analyse transitoire, états logiques de l'inverseur (N_1 et N_2) et tensions sur les noeuds (V_1 et V_2).

Inversement, nous pouvons nous poser la question suivante: le langage VHDL est-il suffisamment puissant pour modéliser une partie analogique avec assez de détails afin de valider un design contenant de l'électronique analogique et numérique? Cette approche est présentée par Dewey [Dew97]. Elle permet dans une certaine mesure de modéliser des parties analogiques. Avec cette approche, il est relativement facile de modéliser des systèmes non-conservatifs au niveau des noeuds (exemple: figure 3-6). Il est moins facile de modéliser un système conservatif au niveau des noeuds tel qu'un schéma électrique classique. Dans les deux cas, il est possible de passer du domaine continu au domaine discret (numérique) par une transformation. Par exemple: approximation du quotient différentiel par un quotient de différences (3.4) (voir § A.1), ou par la transformation bilinéaire (3.5).

$$s \rightleftharpoons \frac{1}{T} \cdot (1 - z^{-1}) \qquad (3.4)$$

$$s \rightleftharpoons \frac{2}{T} \cdot \frac{z-1}{z+1} \qquad (3.5)$$

Les avantages de cette approche sont la rapidité des simulations obtenues et l'utilisation d'un seul simulateur. Les désavantages de cette approche sont les suivants:

- une perte de précision des simulations de la partie analogique;
- il faut valider la modélisation de la partie analogique avec un modèle de type SPICE;
- seules des analyses transitoires peuvent être entreprises;
- la modélisation par des schémas conservatifs au niveau des noeuds est difficile.

Dans ce travail, à cause de ces désavantages et de l'émergence de simulateurs multi-modes de plus en plus performants, nous n'avons pas suivi cette approche.

La simulation multi-mode (électronique analogique et numérique) permet l'interaction de modèles numériques et analogiques. Pour que cela fonctionne, il faut que les transitions d'un monde à l'autre soient précisément définies. Dans le cas de HDL-A ou VHDL-AMS, c'est le langage qui permet le passage d'un monde à l'autre [Mos97b]. Une autre approche consiste à placer des modèles de convertisseurs analogique/numérique (A/N) et numérique/analogique (N/A) aux frontières des deux mondes. Pour simuler ce genre de schéma, il faut que le simulateur permette les deux types de simulation. Ce n'est pas trivial, le simulateur analogique doit accepter des discontinuités provenant de la partie numérique et détecter des passages de seuils à ses sorties connectées aux entrées numériques (voir résultats de l'analyse transitoire figure 3-7).

Notons que la conversion de signaux analogiques en signaux numériques par un convertisseur A/N est une opération qui n'est pas linéaire. Elle est quasi linéaire en admettant une certaine approximation due à la quantification. Le manque de théorie générale traitant la non-linéarité nous oblige à aborder le problème de la quantification comme un problème de bruit. Ainsi la différence

entre un signal analogique et le même signal quantifié (fig. 3-8 a) représente le *bruit de quantification* (fig. 3-8 b).

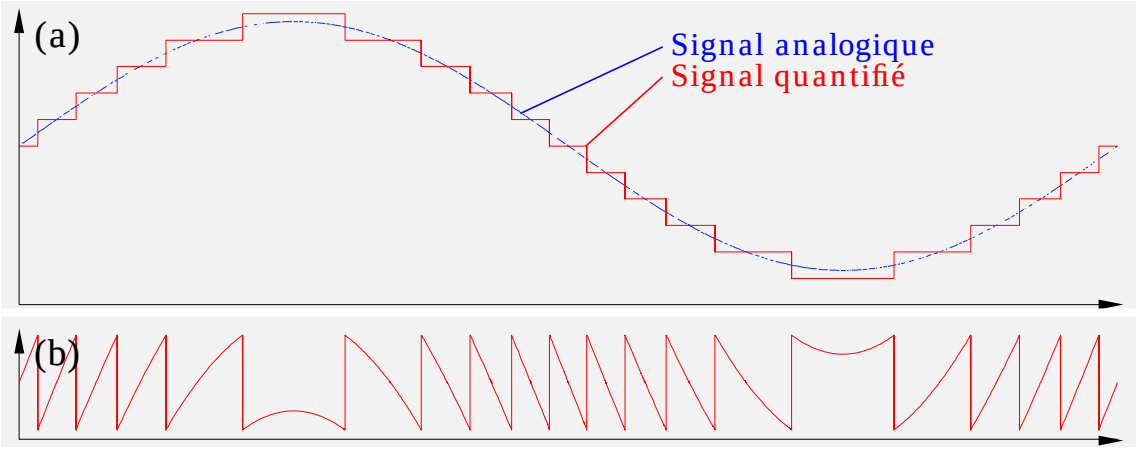


Figure 3-8. (a) Signal analogique, signal quantifié et (b) bruit de quantification.

3.4 Microstructures

Les microstructures qui nous intéressent sont des transducteurs de petites dimensions, par exemple: électromécaniques, optoélectroniques, etc. Il s'agit donc d'éléments hétérogènes touchant à de multiples domaines. Leurs signaux d'entrées et de sorties sont de natures différentes. Des systèmes d'équations différentielles partielles (*PDE*: Partial Differential Equation) et algébriques (*PDAE*: Partial Differential and Algebraic Equation) permettent de décrire le comportement de ce type d'éléments. Une PDE est une équation impliquant une fonction et ses dérivées partielles. L'équation d'onde pour un milieu isotrope est un exemple d'une PDE:

$$\frac{\partial^2 \psi}{\partial t^2} \cdot \frac{1}{v^2} = \frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} + \frac{\partial^2 \psi}{\partial z^2} \qquad (3.6)$$

où ψ représente une perturbation (perturbation longitudinale ou transversale) et v la vitesse de propagation de la perturbation.

Généralement, les PDE sont plus difficiles à résoudre de façon analytique que les ODE. Elles peuvent parfois l'être en utilisant par exemple la fonction de Green, la transformation intégrale ou

une séparation des variables. Mais le plus souvent ce sont les méthodes numériques qui permettent d'obtenir des résultats.

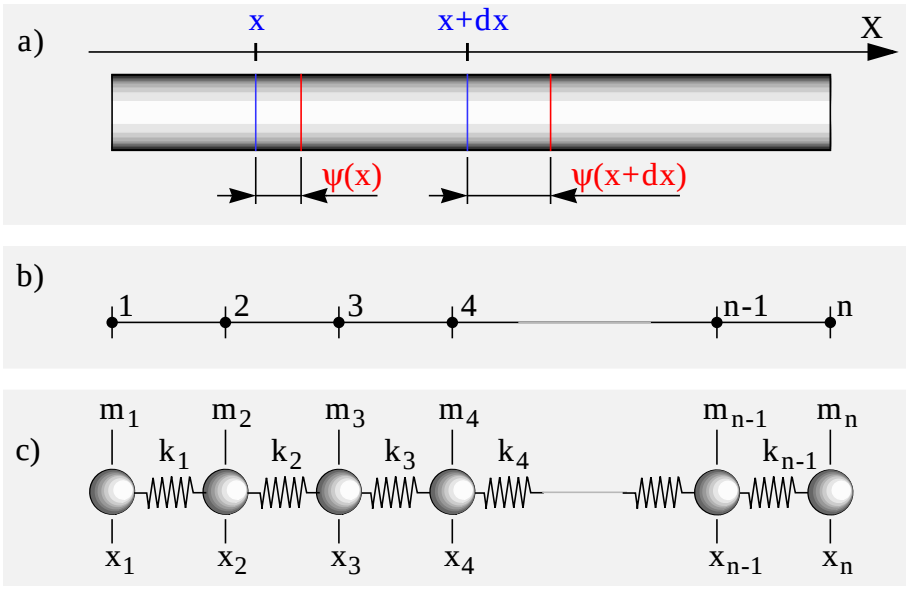


Figure 3-9. (a) Barreau élastique dans lequel se propage une perturbation longitudinale ψ. (b) Décomposition en éléments finis, (c) auxquels on attribue des rigidités et des masses.

Une autre approche, appelée *méthode des éléments finis* (*FEM*: Finite Element Method) [Imb79][Zie73], permet de modéliser le continuum par des éléments discrets appelés *éléments finis* (*FE*: Finite Elements). Le concept d'élément fini suppose le milieu continu divisé en éléments reliés uniquement en un nombre fini de points nodaux. Sur les points nodaux sont appliquées des forces fictives représentant la distribution des contraintes qui s'exercent effectivement aux frontières de l'élément. Les noeuds et éléments n'ont pas forcément de signification physique particulière mais sont basés sur des considérations de précision de l'approximation. Cette méthode permet de transformer un système à constantes réparties (décrit par des PDAE) en un système à constantes localisées (décrit par des ODAE). Le problème se ramène alors à un calcul de structure classique susceptible d'un traitement analytique ou numérique.

Pour illustrer cette approche, considérons le cas d'un barreau élastique dans lequel se propage une perturbation (longitudinale) le long de l'axe X (fig. 3-9 a). En admettant que la perturbation se

propage uniquement le long de l'axe X, le comportement de celle-ci est décrit par l'équation d'onde (3.7).

$$\frac{\partial^2 \psi}{\partial t^2} \cdot \frac{1}{v^2} = \frac{\partial^2 \psi}{\partial x^2} \qquad (3.7)$$

En introduisant la section S, la densité ρ et la tension mécanique σ dans l'équation d'onde (3.7), nous obtenons (voir A.2 page 141) l'équation (3.8).

$$\underbrace{S \cdot dx \cdot \rho}_{\text{masse}} \cdot \underbrace{\frac{\partial^2 \psi}{\partial t^2}}_{\text{accélération}} = \underbrace{dF}_{\text{somme des forces}} \qquad (3.8)$$

où nous reconnaissons la loi du mouvement de Newton (masse multipliant l'accélération est égale à la somme des forces).

En appliquant le concept des éléments finis, nous pouvons diviser le barreau en n éléments (fig. 3-9 a & b). A chaque élément sont attribuées une rigidité et une masse. En appliquant la loi du mouvement à chacune des masses, nous obtenons le système d'ODE (3.9).

$$M \cdot \ddot{\hat{x}} = K \cdot \hat{x} + \hat{F} \qquad \& \qquad \ddot{\hat{x}} = \frac{d^2}{dt^2}\hat{x} \qquad (3.9)$$

où M est une matrice de masses et K une matrice de rigidités (voir § A.2). Ce système d'ODE peut ensuite être traité analytiquement, en utilisant par exemple la transformation de Laplace, ou numériquement, en utilisant un simulateur.

Pour modéliser et simuler des microstructures grâce à la FEM, des outils ont été développés [Aba99] [Ans99] [Mem99]. Les simulateurs de modèles de type FE (*simulateurs FE*) exécutent différentes analyses. Comme pour les simulateurs de type SPICE, les simulateurs FE acceptent généralement les analyses DC, AC et transitoires. Contrairement aux mondes de l'électronique analogi-

que ou numérique, il n'existe pas de langage normalisé pour créer des modèles de type FE.

Grâce à la FEM, nous obtenons une représentation en réseau semblable à ce que nous avons l'habitude de voir en électronique analogique. En effet, considérons le schéma électrique formé de capacités et d'inductances de la figure 3-10. Si nous écrivons les équations caractéristiques de ce schéma (voir § A.2), nous obtenons (3.10).

$$M \cdot \ddot{\vec{U}} = K \cdot \vec{U} + \dot{\vec{I}} \qquad (3.10)$$

où M et K sont exactement les mêmes matrices que dans l'expression précédente. Nous constatons que nous pouvons faire correspondre aux tensions et aux dérivées temporelles du courant, respectivement les positions des masses et les forces appliquées au modèle FE. Ainsi une simulation du circuit électrique avec un simulateur de type SPICE nous donnera le comportement du modèle FE du barreau.

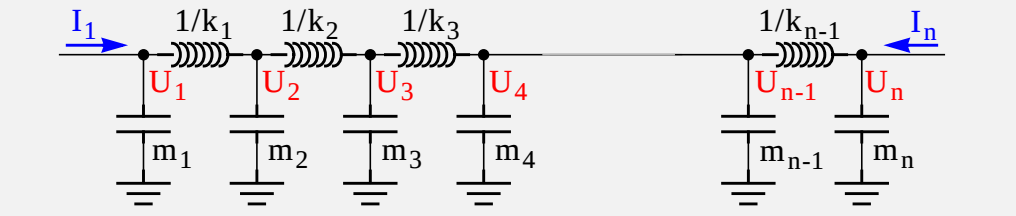


Figure 3-10. Schéma électrique correspondant à la décomposition FE de la figure 3-9 b. U_i sont les tensions des n noeuds, I_1 et I_n sont les courants injectés aux noeuds 1 et n.

Le modèle du barreau que nous venons de présenter est très simple. Nous avons admis que la perturbation se propage uniquement le long de l'axe X. La méthode des FE permet de compliquer le modèle en effectuant un maillage du volume.

Les microstructures utilisées dans les microsystèmes sont généralement plus compliquées que l'exemple du barreau. De plus, le maillage de celles-ci peut être très délicat à établir. Pour de tels cas et les modèles comportant une grande quantité de noeuds, il est difficile de se rapporter à un circuit SPICE équivalent. Pour l'analyse de microstructures, il est alors préférable d'utiliser un

outil dédié à la modélisation et à la simulation FE. Après analyse et dimensionnement d'une microstructure avec un simulateur utilisant la FEM, il est toujours possible d'effectuer une extraction de paramètres que nous pouvons par la suite injecter dans un modèle analytique utilisable avec un simulateur de type SPICE (figure 4-3).

Ainsi, pour répondre au besoin de l'approche MicroSIM, nous pouvons modéliser les microstructures par des modèles à constantes localisées. Ces modèles possèdent des bornes destinées à établir la connexion avec d'autres modèles. Nous distinguons deux types de bornes: celles caractérisées par une seule grandeur et celles caractérisées par deux grandeurs. Le second type permet de créer des réseaux de Kirchhoff dits conservatifs au niveau des noeuds.

Il est possible d'associer aux deux grandeurs caractérisant les noeuds des grandeurs physiques, de manière à ce que la multiplication de ces deux grandeurs donne une puissance. C'est justement le cas en électronique. Ce type de modélisation est basé sur l'énergie.

Pour différents domaines et pour une modélisation basée sur l'énergie, la table 3-1 nous montre les deux grandeurs physiques associées aux noeuds (ou bornes) ainsi que des éléments de base. Pour les mouvements rectilignes ou rotatifs on trouve les deux associations suivantes: vitesse-courant, force-potentiel ou vitesse-potentiel, force-courant (fig. 3-10). Les grandeurs associées au potentiel et au courant peuvent être interverties. Pour ces deux associations, on parle respectivement, d'analogies force-tension (*FV*) ou force-courant (*FI*).

Remarquons qu'il n'est pas obligatoire d'utiliser une modélisation basée sur l'énergie. Pour les mouvements rectilignes, il est par exemple possible d'associer aux noeuds les deux grandeurs suivantes: position et force. La multiplication de ces deux grandeurs ne donne plus une puissance mais une énergie.

Il est donc possible d'utiliser un simulateur de type SPICE pour simuler le comportement de microstructures. Mais, avant d'entreprendre une telle démarche, il est important de se rendre compte que chaque simulateur est développé sur la base de besoins particuliers. Ainsi, nous devons nous demander comment un simulateur manipule d'énormes systèmes d'équations et comment il se comporte lorsqu'il éprouve des difficultés. Par exemple lors de comportements discontinus ou lors de problèmes de convergence.

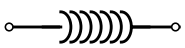
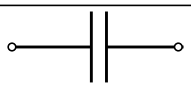
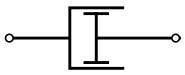
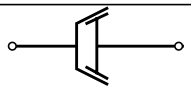
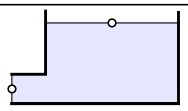
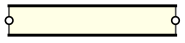
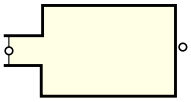
	Noeuds	***Dissip.***	***Retard***	***Accumul.***
Générique	x(t) y(t)	 $x(t)=a\ y(t)$	 $b\ \dot{y}(t)=x(t)$	 $c\ \dot{x}(t)=y(t)$
Electrique	V(t): tension I(t): courant	 Résistance	 Inductance	 Capacité
Mouvement rectiligne	v(t): vitesse F(t): force	 Dissipation	 Elasticité	 Inertie
Mouvement rotatif	ω(t): vitesse M(t): moment	 Dissipation	 Elasticité	 Inertie
Hydraulique	p(t): pression Q(t): débit	 Résistance	 Inertie	 Capacité
Pneumatique ou acoustique	p(t): pression Q(t): débit	 Résistance	 Inertie	 Capacité

Table 3-1. Pour différents domaines et pour une modélisation basée sur l'énergie, grandeurs physiques associées aux noeuds ainsi que quelques éléments de base (primitives).

Les simulateurs d'électronique analogique, tels que les simulateurs de type SPICE, sont optimisés pour manipuler de grandes quantités de transistors (jusqu'à 100'000) alors que les comportements rencontrés présentent rarement des discontinuités.

Les systèmes hydromécaniques (hydraulique avec des valves mécaniques) sont caractérisés par des rigidités très élevées et peu d'amortissement. Le comportement de ce genre de système présente souvent des modes stationnaires de faible fréquence. Lors de changements abrupts, par exemple lors d'un étranglement d'un conduit, des discontinuités importantes apparaissent. On conçoit aisément que les simulateurs développés pour les systèmes hydromécaniques ont des caractéristiques bien différentes des simulateurs de l'électronique analogique.

Pour remédier à ce problème, et dans le but de simuler un circuit électronique connecté à une microstructure, des travaux ont permis de coupler un simulateur FE à un simulateur de type SPICE [Ecc96]. L'approche est intéressante mais extrêmement lourde et les temps de simulation sont importants. Nous ne l'avons pas retenue.

En conclusion, il est possible de créer des macro-modèles ou des modèles HDL-A de microstructures, mais il faut être conscient qu'un simulateur développé pour un domaine ne peut être entièrement satisfaisant pour un autre domaine. C'est l'approche que nous avons suivie dans ce travail. Appuyé par l'arrivée de la norme VHDL-AMS, il nous semble que c'est la voie la plus prometteuse.

Effectivement, les buts visés par VHDL-AMS sont très ambitieux puisqu'il s'agit du premier langage normalisé autorisant la *modélisation multi-niveaux, multi-natures* et *multi-modes*. A terme, il est probable que VHDL-AMS remplace SPICE ainsi que tous les langages tels que HDL-A. Actuellement (2000) et à notre connaissance, il n'existe pas de simulateur supportant entièrement cette nouvelle norme. Le *simulateur multi-domaine* n'existe donc pas encore.

3.5 Software

Selon notre définition, un microsystème remplit une fonction intelligente. Cette fonction peut prendre la forme d'un ou plusieurs *programmes* (SW) fonctionnant avec un ou plusieurs *processeurs* (HW). Du fait de sa flexibilité, cette approche est particulièrement intéressante pour les microsystèmes. Les différents types de processeurs utilisables dans les microsystèmes sont: le *microprocesseur*, le *microcontrôleur* et le *DSP* (Digital Signal Processor). Les définitions de ces trois types de processeurs que nous avons retenues sont:

- Selon Nicoud [Nic86], "*le microprocesseur est au sens strict un circuit intégré unique qui interprète et exécute les instructions d'un programme stocké en mémoire*". Un microprocesseur est donc une unité centrale (*CPU*: Central Processing Unit) placée sur un seul circuit intégré. Une CPU est généralement constituée d'une unité de contrôle (control unit), d'une unité arithmétique et logique (*ALU*: Arithmetic and Logic Unit) et éventuellement d'un ou plusieurs types de mémoires (registres, antémémoire, RAM, ROM).
- Un microcontrôleur est un circuit intégré contenant une CPU, une certaine quantité de RAM et de ROM, des compteurs (counter, timer) et des portes d'entrées/sorties (*I/O ports*). Les microcontrôleurs sont conçus pour fonctionner dans des systèmes embarqués comme par exemple, les microsystèmes.
- Un DSP est un microprocesseur spécialisé dans le traitement du signal. Par rapport aux microprocesseurs conventionnels, les DSP offrent des possibilités arithmétique et logique étendues.

Le SW ou programme que l'on stocke en mémoire est toujours sous la forme de *code machine*. Il s'agit en effet de la seule représentation qu'un processeur est en mesure de lire et d'interpréter. Le code machine se compose de mots binaires dont la longueur est soit toujours la même (8 bits, 16 bits, 32 bits, etc. dans le cas de la plupart des processeurs modernes de type *RISC*: Reduced Instruction Set Computer), soit variable. Pour les microsystèmes, ce sont le

plus souvent de petits processeurs qui sont recherchés (microprocesseurs ou microcontrôleurs de 8, 16 ou 32 bits).

La programmation des processeurs peut être entreprise avec des langages plus ou moins évolués. On dit qu'un langage de programmation est peu évolué ou bas-niveau lorsqu'il est proche du code machine. A contrario, on dit qu'un langage est évolué ou haut-niveau lorsqu'il est éloigné du code machine. Bien qu'une multitude de langages de programmation existent, nous avons arbitrairement choisi de présenter les trois langages suivants. Il s'agit de trois langages de niveau différents susceptibles d'être utilisés pour programmer des processeurs de microsystèmes.

- Le *langage assembleur* est le langage le plus proche du code machine, puisqu'il s'agit d'une représentation symbolique du code machine. Le programme écrit en langage assembleur est converti en code machine grâce à un outil appelé *assembleur*. Programmer en langage assembleur est une opération relativement lente et encline à de nombreuses erreurs, mais c'est la meilleure façon de tirer le maximum d'un HW.
- Le *C* est un langage de programmation permissif et relativement bas-niveau. Contrairement au langage assembleur qui est différent pour chaque famille de processeur, le langage C a été normalisé par *ANSI* (American National Standards Institute), membre de *ISO* (International Organisation for Standardisation), sous le nom de ANSI C. Il s'agit donc d'un langage stable et bien implanté, proche du langage assembleur. Ce langage est de niveau suffisamment élevé pour permettre la création et l'utilisation de librairies et modules. Pour transformer un programme écrit en C en *code objet* il faut un *compilateur*. Finalement pour lier différents codes objets en code machine exécutable, un éditeur de liens (*linker*) est nécessaire.
- Le *C++* fait partie des langages de programmation orientée-objet les plus utilisés. Il s'agit d'un langage haut-niveau permettant de générer et gérer des programmes complexes. Comme pour le C, un compilateur et un éditeur de liens sont

nécessaires pour transformer un programme C++ en code objet. Dans le cadre de ce travail, nous n'avons pas utilisé ce langage de programmation.

Lorsque l'on décide d'inclure un processeur dans un microsystème, il faut que les outils permettant de transformer un programme écrit dans un langage évolué en code machine soient disponibles.

Pour déboguer et optimiser le code, d'autres outils sont également indispensables. Généralement, les fabricants de processeurs fournissent un assembleur et un outil de débogage. L'outil de débogage est un simulateur d'instructions plus ou moins évolué. Ce type d'outils ne permet généralement pas d'effectuer de réelles analyses transitoires puisqu'il ne permet pas de savoir à quel moment une instruction est réellement exécutée. En effet, à cause des pipelines, une instruction n'est pas forcément exécutée lorsqu'elle est lue en mémoire.

Idéalement, conformément à l'approche MicroSIM, il est utile d'effectuer une co-simulation HW-SW d'un microsystème. C'est-à-dire une simulation des programmes avec les processeurs et les périphériques jusqu'aux capteurs et actionneurs (simulation système). Sur le marché de la CAO, il existe quelques solutions [Men99][Vie99] pour de gros processeurs. Mais les outils existants ne permettent pas facilement d'ajouter d'autres processeurs.

Pour de petits processeurs, il existe quelques solutions plus ou moins satisfaisantes. Une solution intéressante est proposée par Virtual Micro Design [VMD99]. Il s'agit d'un simulateur de microcontrôleur universel (*UMPS*: Universal Microprocessor Program Simulator) ayant la propriété de pouvoir simuler l'environnement extérieur au microcontrôleur.

Dans le cadre de ce travail, nous avons adopté une solution pragmatique [Mos98a]. Il s'agit d'un simulateur d'instruction (*ISS*: Instruction Set Simulator) VHDL couplé à une ROM VHDL et à un système de fenêtres permettant de visualiser le programme (voir § 5.1 et fig. 5-11). Les avantages de cette approche sont les suivants:

- Possibilité d'utiliser des outils existants pour générer et déboguer le code machine.
- Le simulateur d'instruction est utilisable par différents simulateurs VHDL.
- Les simulations sont de réelles analyses transitoires.
- Possibilité d'appliquer l'approche MicroSIM.
- Possibilité d'utiliser des processeurs du marché ou des coeurs de processeurs à intégrer

Le principal désavantage de cette approche est qu'il faut écrire et tester (opération très importante et délicate) un nouveau simulateur d'instruction pour chaque processeur.

3.6 Conclusion

Dans ce chapitre, nous nous sommes intéressés à la physique et aux mathématiques permettant de modéliser les sous-systèmes formant un microsystème. La table 3-2 est un résumé de ce que nous avons vu. Nous avons ainsi constaté que l'électronique analogique ou numérique peut être modélisée comme un système à constantes localisées. Or les systèmes à constantes localisées sont décrits par des systèmes d'ODAE. Pour simuler de l'électronique, nous pouvons donc utiliser un simulateur de type SPICE.

Afin de diminuer les temps de simulation, l'électronique numérique est modélisée de façon plus approximative que l'électronique analogique, en utilisant un HDL tel que VHDL. Dans ce cas, la simulation d'électronique analogique et numérique est effectuée par un simulateur multi-mode. Il s'agit en fait du couplage de deux simulateurs: un simulateur de type SPICE et un simulateur commandé par des évènements.

Les microstructures sont généralement des systèmes à constantes réparties. Or ceux-ci se modélisent avec des systèmes de PDAE. Il n'est pas possible de coder une PDAE avec un HDL tel que HDL-A ou VHDL-AMS. Il n'est donc pas possible de simuler un système de PDAE avec un simulateur de type SPICE. Pour

simuler les microstructures avec un simulateur de type SPICE, il faut approcher les systèmes à constantes réparties par des systèmes à constantes localisées. Cela revient à approcher les systèmes de PDAE par des systèmes d'ODAE que nous pouvons coder avec un HDL tel que HDL-A ou VHDL-AMS. De cette manière, les microstructures peuvent être représentées par un réseau compatible avec ce que nous avons l'habitude de voir en électronique.

	Electr. numérique	***Electr. analogique***	***Micro-structures***	***SW***
Fonctions de transfert	Equ. Bool., FSM, Petri net, transf. en Z	ODAEs, transf. de Laplace	PDAEs, FEM	FSM, Transf. en Z
Langages de modélisation	VHDL, Verilog HDL	SPICE, HDL-A, MAST, VHDL-AMS		Assembleur, C, C++, etc.
Types d'analyse	Transitoire	AC, DC, transitoire	AC, DC, transitoire	Transitoire
Types de signaux	Discontinus	Continus	Discontinus & continus	Discontinus
Méthodes de conception	Desc.	Desc. et asc.	Asc. (desc.)	Desc.

Table 3-2. Résumé des outils, langages de modélisation, etc. des quatre types de sous-systèmes (fig. 2-2) formant un microsystème.

Les processeurs utilisés dans les microsystèmes sont généralement de petits processeurs RISC. Leur programmation peut être entreprise avec des langages plus ou moins évolués. Durant le développement d'un microsystème et conformément à l'approche MicroSIM, il est utile d'effectuer de la co-simulation HW-SW. Dans ce travail, nous avons choisi de simuler les processeurs avec un simulateur commandé par des évènements. Pour y parvenir, nous devons modéliser chaque processeur par un ISS couplé à une

ROM. Cette approche nous permet également d'utiliser le langage VHDL.

3.7 Références

[Aba99] *"ABAQUS, Products at Hibbitt, Karlsson & Sorensen, Inc."*, http://www.hks.com/

[ANA94] *"HDL-A User's Manual"*, ANACAD Electrical Engineering Software, Version v1.4_1, Revision v2.0, Sep., 1994

[Ana99] *"Products & Services at Analogy"*, http://www.analogy.com

[Ans96] Y.Ansel, B.Romanowicz, Ph.Lerch, Ph.Renaud, *"Optimisation of a vibrating gyroscope by system level simulation"*, Eurosensors X, Leuven, Sept. 8-11, 1996

[Boi83] R.Boite, J.Neirynck, *"Théorie des réseaux de Kirchhoff"*, Traité d'électricité, Volume IV, Presses polytechniques romandes, 1983

[Cin99] *"University of Cincinnati, Department of Electrical and Computer Engineering and Computer Science. SEAMS a Simulation Environment for VHDL-AMS"*, http://www.ececs.uc.edu

[Deg94] M.Degrauwe, *"Electronique analogique avancée"*, Cours donné à l'Institut de Microtechnique de l'Université de Neuchâtel, 1994

[Dew97] A.Dewey, *"Applicability of Discrete Event Hardware Description Languages to the Design and Documentation of Electronic Analog Systems"*, Analog and Mixed-Signal Hardware Description Languages, Current Issues in Electronic Modeling, A. Vachoux, J.-M. Bergé, O. Levia, J. Rouillard (eds.), Kluwer Academic Publischers, Dordrecht, NL, pp. 1-17, 1997

[Ecc96] P.-C.Eccardt, M.Knoth, G.Ebest, H.Landes, C.Clauss, S.Wünche, *"Coupled Finite Element and Network Simulation for Microsystem Components"*, MICRO SYSTEM Technologies 96, Potsdam, Sep.17-19, 1996, p.145-150

[Fot97] D.Foty, *"MOSFET Modeling with SPICE"*, Prentice Hall PTR,1997

[IEE93] *"IEEE Standard VHDL Language Reference Manual"*, IEEE Std 1076-1993, The Institute of Electrical and Electronics Engineers Inc., 1993

[IEE95] *"IEEE Standard Hardware Description Language Based on the Verilog® Hardware Description Langage"*, IEEE Std 1364-1995, The Institute of Electrical and Electronics Engineers Inc., 1995

[IEE99a] *"IEEE: Institute of Electrical and Electronics Engineers"*, http://ieee.org/

[IEE99b] *"VHDL Language Reference Manual (Integrated with VHDL-AMS changes)"*, IEEE Std 1076.1-1999, The Institute of Electrical and Electronics Engineers Inc., Approved 18 March 1999

[Imb79] J.F.Imbert, *"ANALYSE des STRUCTURES par ELEMENTS FINIS"*, SUP'AERO, Ecole supérieure de l'aeronautique et de l'espace, Cepadues Editions, 1979

[Kap96] E.D.Kaplan et al., *"Understanding GPS, Principles and Applications"*, Mobile Communications Series, Artech House Publishers, Boston, London, 1996

[Mem99] *"MEMCAD, Products at Microcosm Technologies"*, http://www.memcad.com/

[Men99] *"Products at Mentor Graphics"*, http://www.mentorg.com/

[Mos97b] V. Moser, H.-P. Amann, F. Pellandini, *"Behavioural Modelling of Sampled-Data Systems with HDL-A and ABSynth"*, Hardware description Languages and their Applications, IFIP TC10 WG10.5, International Conference on Computer Hardware Description Languages and their Applications, Toledo, S, April 20-25, 1997, C. Delgado Kloos, E. Cerny (eds.), Chapman & Hall, London, UK, pp. 159-177, 1997

[Mos98a] V. Moser, A. Boegli, H.P. Amann, F. Pellandini, *"VHDL-based HW/SW Cosimulation of Microsystems"*, FDL'98, Forum on Design Languages '98, Lausanne, Switzerland, Sep. 6-11, 1998, p.157-163

[Nag75] L.W.Nagel, *"SPICE2: A Computer Program to Simulate Semiconductor Circuits"*, ERL Memo ERL-M520, University of California, Berkeley, 1975

[Nic86] J.-D.Nicoud, *"Calculatrices"*, Traité d'électricité, Volume XIV, Presses polytechniques romandes, 1986

[Pis80] N.Piskounov, *"Calcul différentiel et intégral"*, Tome I et II, Editions Mir de Moscou, 1980

[Vie99] *"Products at ViewLogic"*, http://www.viewlogic.com/

[VMD99] *"Products at Virtual Micro Design"*, http://www.vmdesign.com/

[Zie73] O.C.Zienkiewicz, *"La Méthode des Eléments Finis, Appliquée à l'art de l'ingénieur"*, Ediscience, Paris, 1973

Chapitre 4

Développement de modèles

L'approche MicroSIM est une méthode basée sur des outils de CAO. Pour que cette approche soit applicable, il faut que des design-kits (c.f. § 2.3) existent et soient utilisables. L'environnement que nous avons choisi pour ce travail, est un environnement développé pour la CAO d'électronique. Les simulateurs que nous utilisons ont donc été optimisés en conséquence et les bibliothèques de modèles de microstructures sont peu nombreuses. Afin d'appliquer l'approche MicroSIM dans l'environnement choisi, nous devons développer un design-kit contenant des modèles de microstructures. Or, développer des modèles de microstructures dans un tel environnement demande de l'expérience. Remarquons par exemple, que tout modèle dévoile uniquement certains aspects de la réalité avec plus ou moins de détails. Un "bon modèle" est un modèle suffisamment complexe, dévoilant juste l'aspect désiré de la réalité et donnant de "bons résultats" de simulation.

La première partie de ce chapitre présente une version adaptée du diagramme Y de Gajski et Kuhn. Celui-ci permet d'énumérer, avec différents niveaux d'abstraction, les formes comportementales, les composants structurels et les objets physiques d'un microsystème ou sous-système. Ensuite nous présentons le design-kit pour la simulation globale de microsystèmes, que nous avons créé afin d'appliquer l'approche MicroSIM dans l'environnement de CAO choisi. La suite du chapitre présente des exem-

ples de modèles. Ceux-ci ont été choisis afin d'illustrer les difficultés rencontrées lors de leur élaboration.

4.1 Points de vues et niveaux d'abstraction

Pour concevoir un microsystème, différents spécialistes doivent collaborer. Or, chacun d'eux a un point de vue particulier et, pour réaliser son travail, a besoin d'informations spécifiques. Il est possible de distinguer trois points de vue accentuant:

- soit l'aspect comportemental,
- soit l'aspect structurel,
- soit l'aspect physique.

Une *représentation comportementale* est une boîte noire spécifiant le comportement des sorties en fonction des entrées et du temps. Ce type de représentation ne donne aucune information sur l'implémentation de la fonctionnalité. La *représentation structurelle* permet de définir la boîte noire en matière de composants et de connexions. Cette représentation ne décrit pas explicitement la fonctionnalité. Finalement la *représentation physique* spécifie les caractéristiques physiques (dimensions, localisations, etc.) des composants de la représentation structurelle. Ainsi la représentation structurelle décrit les connexions entre les éléments, alors que la représentation physique décrit les relations spatiales entre les composants.

Les trois représentations ou points de vue peuvent être plus ou moins détaillés. Le *diagramme Y* de Gajski et Kuhn [Gaj83] permet d'énumérer, avec différents niveaux d'abstraction, les *formes comportementales*, les *composants structurels* et les *objets physiques* d'un système numérique. La figure 4-1 nous montre un tel diagramme adapté aux microsystèmes ou sous-systèmes, avec trois niveaux d'abstraction (cercles concentriques):

- *fonctionnel*,
- *comportemental*,
- *primitif*.

Remarquons qu'il est possible de définir plus de niveaux, mais qu'il est difficile de les nommer sans les associer à l'un des trois points de vue!

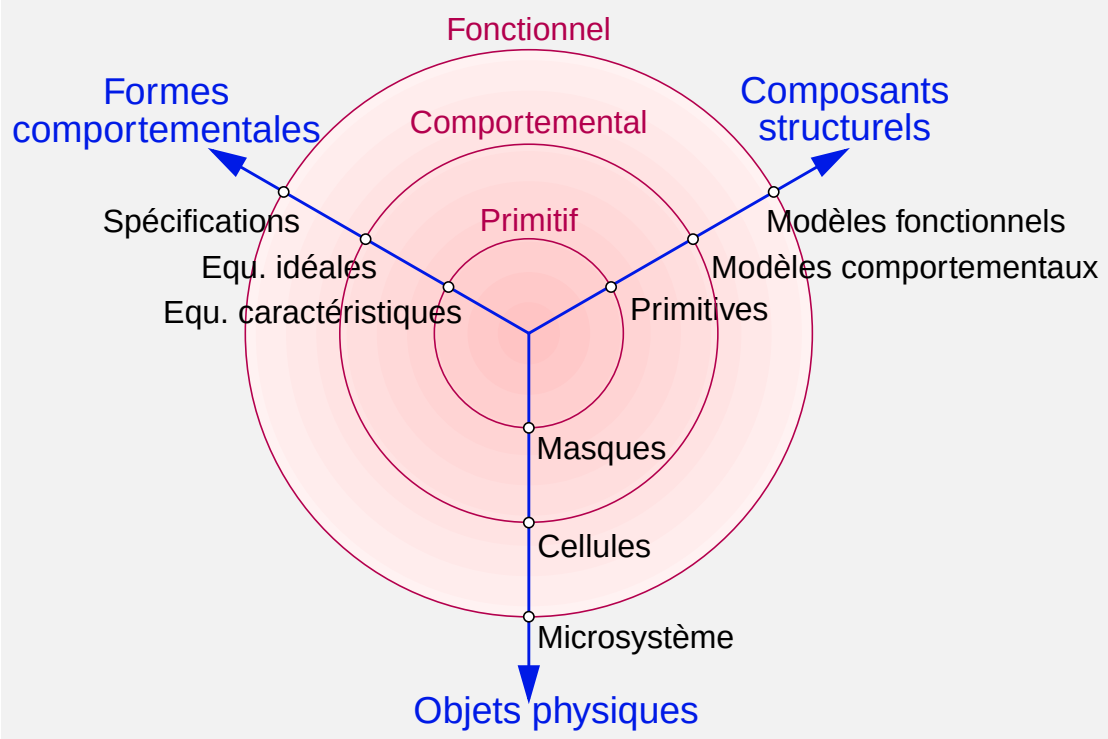


Figure 4-1. Diagramme Y adapté au cas des microsystèmes

Le niveau fonctionnel est le niveau le plus haut contenant le plus d'abstractions. Ce niveau permet de décrire des systèmes complexes avec peu de détails, par exemple sous la forme d'un diagramme de bloc. Le niveau primitif est le niveau le plus bas. A ce niveau, le microsystème est décrit par un schéma interconnectant des primitives (composants structurels). Les primitives sont décrites par des équations caractéristiques (formes comportementales). Les objets physiques de ce niveau sont les masques permettant de réaliser le microsystème. Entre le niveau fonctionnel et primitif, il est possible d'en ajouter d'autres. Sur la figure, nous en avons représenté un seul, il s'agit du niveau comportemental.

Dans le cadre de la conception descendante, le designer débute son travail au niveau du cercle le plus grand. Ensuite, il effectue les opérations de décompositions afin de descendre vers le centre du cercle. Au besoin, durant les opérations de décomposition, il passe d'un point de vue à l'autre. Généralement le travail débute

avec une forme comportementale au niveau fonctionnel, pour passer à un moment donné par une représentation structurelle et finir avec des objets physiques au niveau primitif [Wal85].

Dans le chapitre 3, nous avons passé en revue les langages et simulateurs autorisant la modélisation de microsystèmes. Il s'agit d'outils permettant de modéliser et simuler des formes comportementales de parties de microsystèmes. Ces outils permettent également de simuler des composants structurels connectés entre eux, mais pas de manipuler des objets physiques. Ainsi, dans ce qui suit, nous nous restreignons aux formes comportementales et aux composants structurels.

4.2 "GLOBS Microsystem Design Kit"

L'environnement de CAO que nous avons choisi pour réaliser des simulations globales de microsystèmes est un environnement développé pour la CAO d'électronique numérique et analogique. Il s'agit de l'environnement de Mentor Graphics® [Men99]. Le principal avantage de cet environnement est qu'il possède plusieurs simulateurs. Ceux-ci autorisent la simulation de modèles analogiques tels que SPICE ou HDL-A, et numériques tels que VHDL. Un autre avantage est que, pour cet environnement, de nombreux design-kits existent et permettent de développer des circuits électroniques. En revanche, des bibliothèques de modèles de microstructures sont peu nombreuses.

Dans le cadre de ce travail, nous avons développé et rassemblé des modèles d'éléments pouvant entrer dans la composition d'un microsystème. Nous nous sommes en particulier intéressés aux modèles:

- de microstructures,
- de sources permettant de créer des bancs de tests,
- de quantificateurs permettant l'étude d'implémentations d'algorithmes,
- de convertisseurs A/N,

- d'ISS (Instruction Set Simulator) de microcontrôleurs.

Le système de fenêtres permettant de visualiser un programme en assembleur (co-simulation HW-SW) a également été développé de manière à fonctionner avec l'environnement choisi. Nous avons appelé cet assemblage design-kit pour la simulation globale de microsystèmes (*GLOBS Microsystem Design Kit*. *GLOBS*: GLOBal Simulation). Celui-ci est disponible sous la forme d'un CD-ROM:

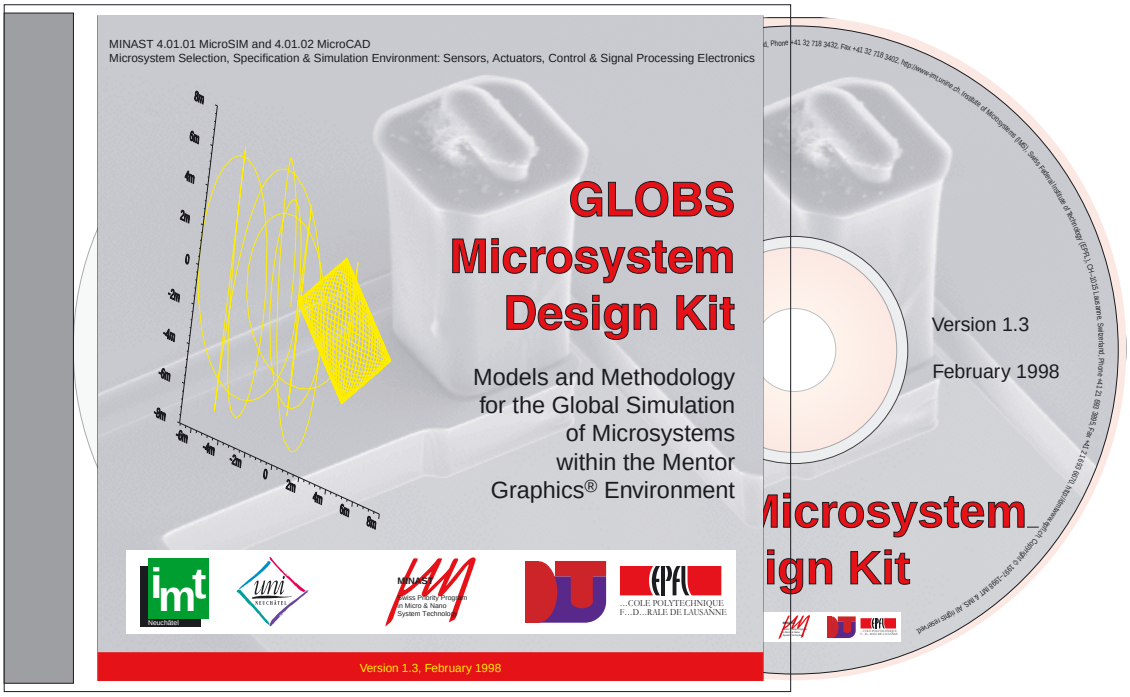


Figure 4-2. "GLOBS Microsystem Design Kit" disponible sous la forme d'un CD-ROM.

Ce CD-ROM contient également un document formé de trois chapitres. Le premier est un guide d'installation du design-kit. Le deuxième est dédié à la co-simulation HW-SW. Le troisième est formé de la liste et de la documentation des modèles. La documentation d'un modèle contient:

- une description succincte du modèle,
- le type du modèle (VHDL, SPICE, HDL-A, etc.),
- son symbole (avec les bornes),
- la liste et description des bornes,
- la liste et description des paramètres,

- une description détaillée du modèle,
- un exemple de simulation (description, schémas et résultats),
- des références,
- une liste de modèles apparentés.

Dans sa version actuelle (version 1.4), ce design-kit possède un peu plus d'une trentaine de modèles (c.f. annexes B). Ce travail constitue une première ébauche de design-kit. Il doit être perçu comme une réponse aux besoins de l'approche MicroSIM.

Certains des modèles de microstructures se trouvant dans ce design-kit nous ont posé des problèmes. Soit lors de leur élaboration, soit lors de simulations. Dans ce qui suit (§ 4.3), nous nous sommes efforcés de présenter des exemples de modèles de microstructures révélant un certain nombre de problèmes rencontrés.

4.3 Exemples de modèles de microstructures

Pour que l'approche MicroSIM soit applicable, il faut que des bibliothèques de microstructures existent et soient utilisables. Pour remplir ces bibliothèques, différentes voies sont envisageables (fig. 4-3).

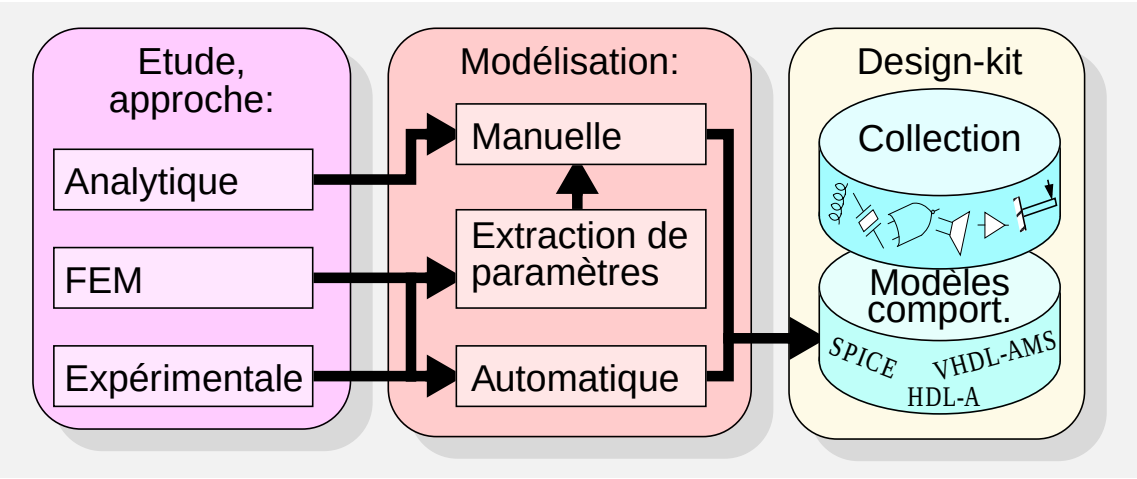


Figure 4-3. Approches permettant de remplir une bibliothèque de modèles de microstructures.

Avant de modéliser une microstructure, il faut l'étudier. Cette première étape peut être entreprise de différentes manières. Nous en distinguons trois: analytique, éléments finis (FE) et expérimental (confection de prototypes que l'on caractérise). Notons que ces différentes approches sont souvent complémentaires. Il est par exemple possible de créer un modèle en partant de formules physiques (approche analytique) dont les paramètres (constantes d'élasticité, amortissement, etc.) sont déterminés par simulation FE ou expérimentalement (extraction de paramètres). Pour ce travail, nous nous sommes restreints aux approches analytique et expérimentale. Pour commencer, voyons l'exemple d'un capteur de température.

4.3.1 Capteur de température.

Comme premier exemple de modélisation, nous avons choisi une diode à jonction que nous désirons utiliser comme capteur de température. Pour modéliser cet élément, nous adoptons l'approche analytique. Les formes comportementales du niveau comportemental (diagramme Y fig. 4-1) sont définies par les équations idéales (4.1) & (4.2) [Asc82]. Il s'agit des équations de la diode théorique.

$$I_D(T, U_D) = I_S(T) \cdot \left(e^{\frac{q \cdot U_D}{k \cdot T}} - 1 \right) \qquad (4.1)$$

où q est la charge élémentaire, k la constante de Boltzmann, T la température absolue et I_S le courant inverse de saturation. Notons que le courant inverse de saturation dépend également de la température absolue:

$$I_S(T) = \kappa \cdot T^{\eta} \cdot e^{-\frac{q \cdot V_G}{k \cdot T}} \qquad (4.2)$$

où V_G est la hauteur de la bande interdite et où κ et η sont des constantes indépendantes de T. Les équations de la diode idéale définissent une surface (fig. 4-4 b).

Il est possible d'inverser la relation (4.1) afin d'obtenir une expression nous donnant la tension sur la diode en fonction du courant injecté et de la température de la diode (4.3).

$$U_D(T, I_D) = \frac{k \cdot T}{q} \cdot \ln\left(\frac{I_D}{I_S(T)} + 1\right) \qquad (4.3)$$

Le symbole de cette modélisation possède trois bornes (fig. 4-4 a). Deux sont des bornes électriques caractérisées par deux grandeurs (tension et courant). La troisième est une borne caractérisée par une seule grandeur: la température.

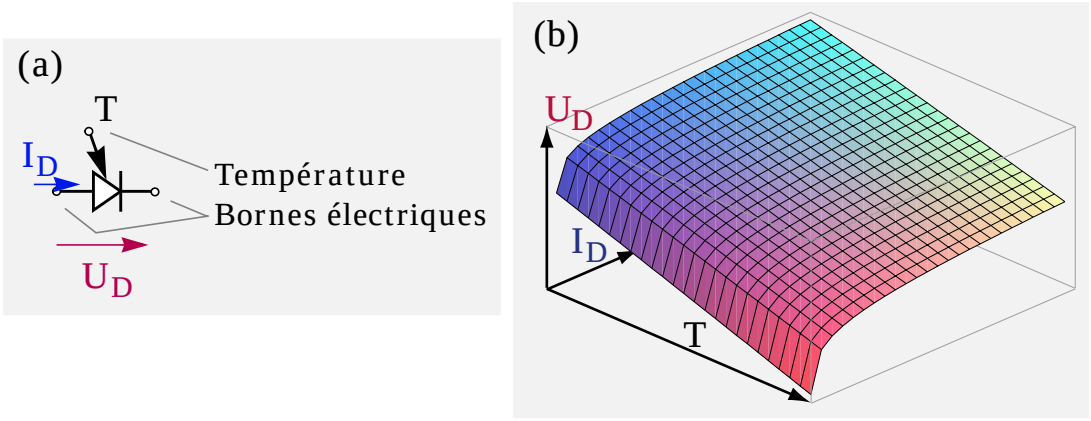


Figure 4-4. (a) Symbole de la diode utilisée comme capteur de température. (b) Surface définie par les équations de la diode idéale utilisée dans le sens direct.

Une des difficultés de la modélisation analogique en réseau de Kirchhoff est que l'on a parfois de la peine à distinguer les entrées et les sorties des modèles. Dans le cas de la diode utilisée comme capteur de température, la borne de la température est toujours une entrée. Mais, aux bornes électriques de ce symbole, il est envisageable de fixer une tension et d'utiliser le courant comme mesure de la température. Dans ce cas, la température peut être considérée comme l'entrée du système, alors que le courant traversant la diode peut être considéré comme la sortie du système. La figure 4-5 a nous montre le schéma électrique, le symbole de type "bloc" et la réponse en température de ce choix.

Un autre choix consiste à faire le contraire, c'est-à-dire fixer le courant et utiliser la tension comme mesure de la température

(fig. 4-5 b). Il est également envisageable de créer un montage (fig. 4-5 c) où le courant et la tension varient avec la température.

Le symbole de la figure 4-4 a est une vue externe du modèle, il s'agit d'un composant structurel (cf. diagramme Y fig. 4-1). Avec le langage HDL-A, la vue interne de ce modèle peut être entièrement codée dans un bloc RELATION (fig. 3-4) en utilisant soit les équations (4.1) et (4.2), soit les équations (4.3) et (4.2). Théoriquement, ces deux façons de coder la vue interne du modèle sont strictement identiques. Dans le premier cas, le modèle est codé comme si la tension sur la diode était imposée et le courant calculé. Dans le deuxième cas c'est le contraire, c'est comme si c'était le courant qui était imposé et la tension calculée.

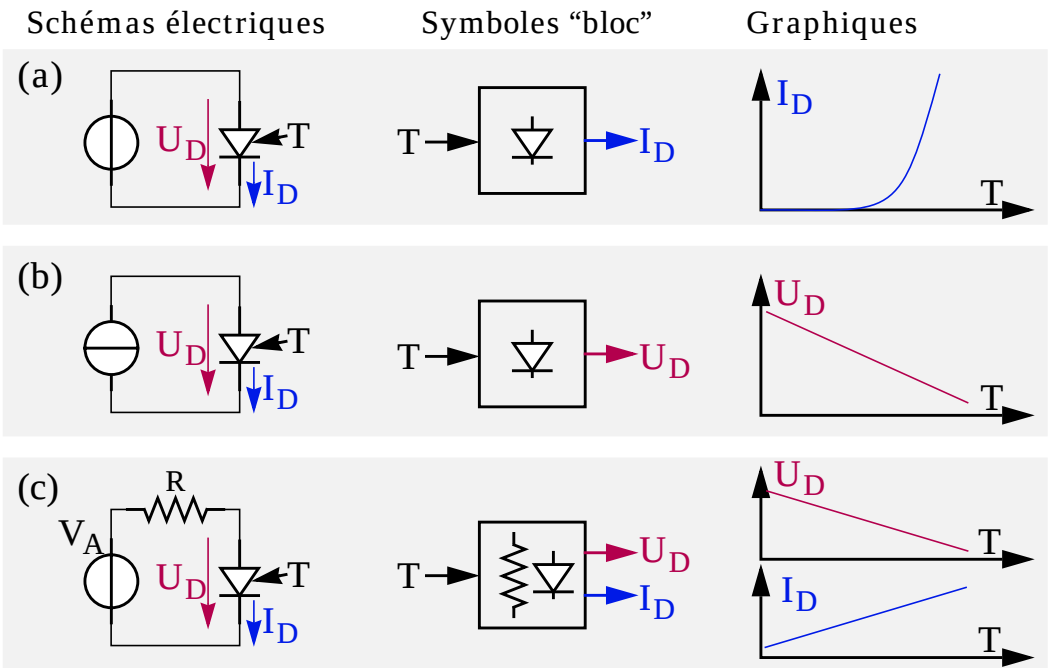


Figure 4-5. Schémas électriques, symboles "blocs" et réponses graphiques pour différentes façons de mesurer la température: (a) le courant est la mesure de la température, (b) la tension est la mesure de la température, (c) le courant et la tension sont des mesures de la température.

Pratiquement, nous avons pu constater que ces deux modèles ne sont pas identiques. Si nous imposons un courant au modèle codé de la première manière, nous constatons que le simulateur (Accusim II, c.f. § B.1) ne converge pas, aucune solution n'est obtenue. Ce genre de problèmes peut parfois être résolu en jouant avec

les paramètres du simulateur. Mais c'est souvent au détriment de la rapidité de la simulation. Nous retenons de cet exemple que:

- un modèle analogique même simple, peut poser des problèmes de convergence,
- avec certains langages et en particulier HDL-A, la façon de coder les équations a une influence sur les performances des simulations,
- un savoir-faire est nécessaire pour construire de "bons" modèles. Il ne suffit pas de connaître le langage de modélisation, il faut également connaître le simulateur.

4.3.2 Capteur de pression piézo-résistif

Comme second exemple de modélisation, nous avons choisi le capteur de pression piézo-résistif. La *piézo-résistivité*, à ne pas confondre avec la piézo-électricité, est la dépendance de la résistivité de certains cristaux, en particulier celui de silicium, à la contrainte mécanique. Un capteur de pression piézo-résistif est composé d'une membrane de silicium sur laquelle sont diffusées des piézo-résistances (fig. 4-6). Lorsque la membrane est soumise à une différence de pression, les piézo-résistances subissent une contrainte mécanique. Les piézo-résistances sont connectées en pont de Wheatstone. Celles-ci sont diffusées dans la membrane de manière à ce que la tension différentielle du pont soit une mesure de la contrainte. La réponse de ce type de capteur dépend également de la température.

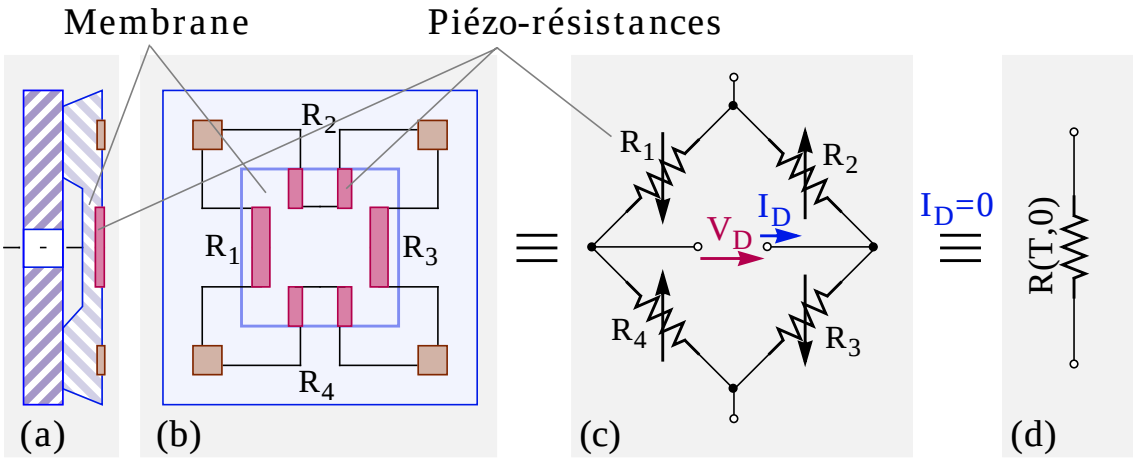


Figure 4-6. Capteur de pression piézo-résistif, (a) vue en coupe, (b) vue de dessus, (c) équivalent électrique, (d) résistance équivalente lorsque ID=0.

Pour modéliser les piézo-résistances, il est possible d'adopter une approche analytique [Rom97c]. En admettant que la membrane est carrée, il est possible d'obtenir les relations (4.4) & (4.5).

$$\left(\frac{w_{11}}{h}\right)^3 + 0.25223 \cdot \frac{w_{11}}{h} = 1.333 \cdot 10^{-4} \cdot p \cdot \frac{a^4}{D \cdot h} \qquad (4.4)$$

$$D = (E + T \cdot K) \cdot \frac{h^3}{12 \cdot (1 - \nu^2)} \qquad (4.5)$$

où w_{11} est le coefficient de déplacement de Fourier, K le coefficient thermique du module de Young, E le module d'élasticité, p la différence de pression subie par la membrane, T la température, D la rigidité en flexion, ν le coefficient de Poisson, h l'épaisseur de la membrane et a la longueur de la membrane.

Les équations (4.4) à (4.5) permettent de calculer w_{11} que l'on peut ensuite injecter dans les relations (4.6) et (4.7). Celles-ci nous donnent les valeurs des piézo-résistances en fonction de la température et de la pression.

$$R_1(T, p) = R_3(T, p) = r_0 \cdot \left[1 - 2.5223 \cdot \frac{r_s}{r_0} \cdot w_{11}(T, p)\right] \qquad (4.6)$$

$$R_2(T, p) = R_4(T, p) = r_0 \cdot \left[1 + 2.5223 \cdot \frac{r_s}{r_0} \cdot w_{11}(T, p) \right] \qquad (4.7)$$

où r_0 est la valeur de la piézo-résistance pour une température donnée et une différence de pression nulle, et r_s la sensibilité de la piézo-résistance à la différence de pression.

La mesure de capteurs de pression piézo-résistifs nous apprend qu'une tension d'offset à la sortie du pont de Wheatstone est toujours présente. Les causes de cet offset sont multiples: appariement des piézo-résistances, conditionnement du capteur dans un boîtier, etc. Nous constatons qu'il est facile de modéliser un mauvais appariement de piézo-résistances; mais il est beaucoup plus difficile d'aborder les effets du conditionnement des capteurs de façon analytique.

En étudiant le modèle présenté précédemment, relations (4.4) à (4.7), nous constatons que la résistance équivalente du pont R(T,0) est égale à une constante r_0. Or la pratique nous montre que ce n'est pas le cas. En effet, on utilise souvent la résistance équivalente du pont pour mesurer la température. Bien qu'analytique, ce modèle n'est pas suffisamment réaliste pour être utilisé dans le développement d'une électronique. Cet exemple nous montre à quel point il est important de valider le modèle d'une microstructure avant de l'utiliser pour le développement d'une électronique.

Pour créer un modèle de capteur de pression piézo-résistif plus réaliste, nous avons adopté une autre approche [Boe95]. Nous nous sommes basés sur la physique (approche analytique) et sur des mesures de capteurs (approche expérimentale). Nous avons ainsi obtenu les relations (4.8) à (4.10).

$$R_1(T, p) = R_3(T, p) = R(T;0) \cdot \left[1 - \sum_{i=1}^{N_P} F_i(T) \cdot p^i \right] \qquad (4.8)$$

$$R_2(T, p) = R_4(T, p) = R(T;0) \cdot \left[1 + \sum_{i=1}^{N_P} F_i(T) \cdot p^i \right] \qquad (4.9)$$

$$r(T) = \frac{2 \cdot R(T;0) \cdot G(T)}{1 - G(T)} \qquad (4.10)$$

où R(T;0) correspond à la résistance équivalente du pont lorsque I_D=0 (fig. 4-6). Les résistances r(t) sont des résistances de petites valeurs permettant de modéliser l'offset. Elles sont ajoutées en série aux résistances R2 et R4 (fig. 4-7 b). Les fonctions $F_i(T)$ correspondent aux termes définissant la sensibilité à la pression, et G(T) correspond à l'offset. Ces différentes fonctions peuvent être modélisées par des polynômes en température (équation (4.11) à (4.13)).

$$R(T;0) = \sum_{i=0}^{N_R} a_i \cdot T^i \qquad (4.11)$$

$$F_j(T) = \sum_{i=0}^{N_F} c_{ji} \cdot T^i \qquad (4.12)$$

$$G(T) = \sum_{i=0}^{N_G} q_i \cdot T^i \qquad (4.13)$$

Le symbole du modèle d'une piézo-résistance possède quatre bornes (fig. 4-7 a). Deux sont des bornes électriques caractérisées par deux grandeurs. Les deux autres bornes sont des entrées et sont caractérisées par une seule grandeur, à savoir la température pour l'une et la pression pour l'autre.

Avec cette modélisation, dans le cas d'un pont alimenté avec une tension constante Va (fig. 4-7 b), la tension différentielle normalisée (V_D/Va) définit une surface (fig. 4-7 c).

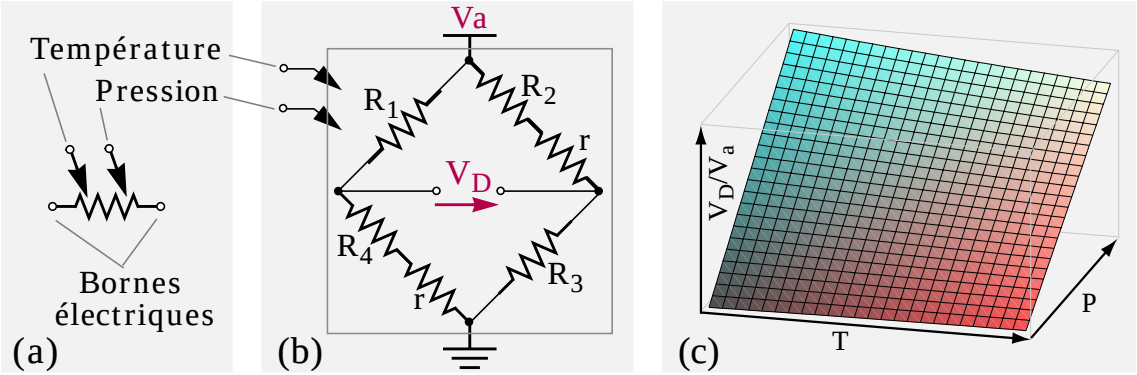


Figure 4-7. (a) Symbole d'une piézo-résistance d'un capteur de pression. (b) Pont de Wheatstone alimenté avec une tension constante. (c) Tension différentielle normalisée en fonction de la température et de la pression.

Afin de valider ce modèle, nous avons mesuré un capteur de pression absolue du commerce [Jos96]. Par régression des mesures, nous avons calculé les constantes des polynômes (équations (4.11) à (4.13)). Nous nous sommes restreints à des polynômes d'ordre 2, c'est-à-dire $N_p=N_R=N_F=N_G=2$.

La figure 4-8 nous montre les mesures obtenues avec le capteur comparées à une simulation (modèle HDL-A). Nous constatons que le modèle décrit la réalité de façon suffisamment réaliste pour être utilisé lors du développement d'une électronique. Cela constitue la phase de validation de ce modèle.

Remarquons que notre modèle de capteur de pression piézo-résistif ne présente pas de comportement fréquentiel élaboré. Dans la mesure où les variations de pression et de température sont lentes, ce modèle est suffisamment réaliste. En conclusion, nous retenons de cet exemple que:

- un modèle purement analytique ne décrit pas forcément la réalité avec suffisamment de détails,
- bien que cela paraisse évident, pour créer un modèle réaliste d'une microstructure, il faut bien la connaître,
- le modèle d'une microstructure doit être validé avant d'être utilisé pour créer une électronique réaliste correspondante,

- la validation d'un modèle est une opération fondamentale et délicate.

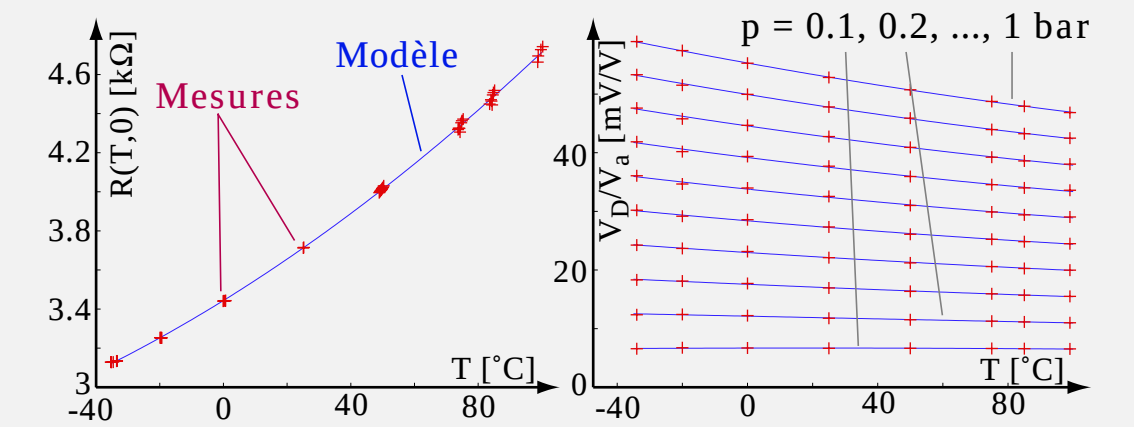


Figure 4-8. Comparaison des mesures et du modèle du capteur de pression MS5201-AP de Micronas. A gauche: résistance équivalente du pont. A droite: tension différentielle normalisée (Va étant la tension d'alimentation du pont).

4.3.3 Capteur actionneur piézoélectrique

Comme troisième exemple, nous avons choisi un élément piézoélectrique. La *piézoélectricité* est la propriété de certains cristaux de produire un courant électrique lorsqu'ils sont soumis à des tensions mécaniques [Ike96]. Ces mêmes cristaux présentent l'effet piézoélectrique inverse, c'est-à-dire une déformation sous l'action d'un champ électrique. Les éléments piézoélectriques peuvent donc être utilisés comme capteurs et comme actionneurs. Nous avons choisi de présenter cet élément car il présente des caractéristiques fréquentielles intéressantes à modéliser.

Pour modéliser un élément piézoélectrique il faut séparer la partie mécanique de la partie électrique (fig. 4-9 a). La partie mécanique est modélisée par un oscillateur harmonique amorti et la partie électrique par une capacité et des résistances. Ces deux parties communiquent par le biais d'un transformateur idéal dont le rapport est ϕ.

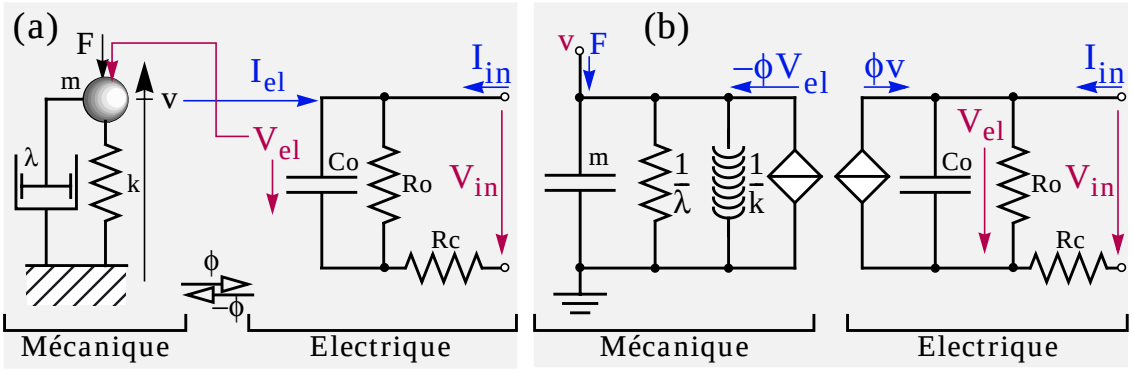


Figure 4-9. Modélisation d'un élément piézoélectrique, (a) schéma de principe, (b) macro-modèle.

Plus précisément, un déplacement du côté mécanique induit un courant du côté électrique alors qu'une tension du côté électrique induit une force du côté mécanique. Du côté électrique, la résistance Rc permet de tenir compte de l'influence d'un défaut de contact aux électrodes. La résistance Ro modélise l'effet de courants de fuites dans l'élément. Dans le domaine de Laplace, les équations de ce système sont les suivantes:

$$F = \left(m \cdot s + \frac{k}{s} + \lambda\right) \cdot v + \phi \cdot V_{el} \qquad (4.14)$$

$$V_{el} = \frac{V_{in} + Rc \cdot \phi \cdot v}{1 + \frac{Rc}{Ro} + Rc \cdot Co \cdot s} \qquad (4.15)$$

$$I_{in} = \frac{V_{in} - V_{el}}{Rc} \qquad (4.16)$$

où F est la force appliquée, v la vitesse de la masse m, k la constante de ressort et λ l'amortissement. Du côté électrique, V_{in} et I_{in} sont la tension et le courant appliqué et Co la capacité de l'élément piézoélectrique.

Il s'agit d'un modèle analytique pouvant être codé avec un langage tel que HDL-A, VHDL-AMS ou sous la forme d'un macro-modèle. La figure 4-9 b nous montre le macro-modèle utilisant l'analogie force-courant (FI) pour la partie mécanique. Nous avons ainsi pu constater que les temps de simulation du macro-modèle

sont bien plus courts que ceux du modèle HDL-A (un facteur 5 pour des simulations AC).

Afin de valider ce modèle, un modèle FE a été confectionné et simulé. Ensuite un élément piézoélectrique a été construit et mesuré. C'est en jouant avec les différents paramètres que nous avons obtenus des simulations conformes aux mesures effectuées. La méthode adoptée est la suivante, tout d'abord, nous avons fixé les *pulsations série* (4.17) et *parallèle* (4.18).

$$\omega_s = \sqrt{\frac{k}{m}} \qquad (4.17)$$

$$\omega_p \cong \omega_s \cdot \sqrt{1 + x} \qquad \& \qquad x = \frac{\phi^2}{k \cdot Co} \qquad (4.18)$$

Ensuite nous avons ajusté l'amortissement mécanique λ (fig. 4-10), puis la résistance parallèle Ro et pour finir la résistance de contact Rc.

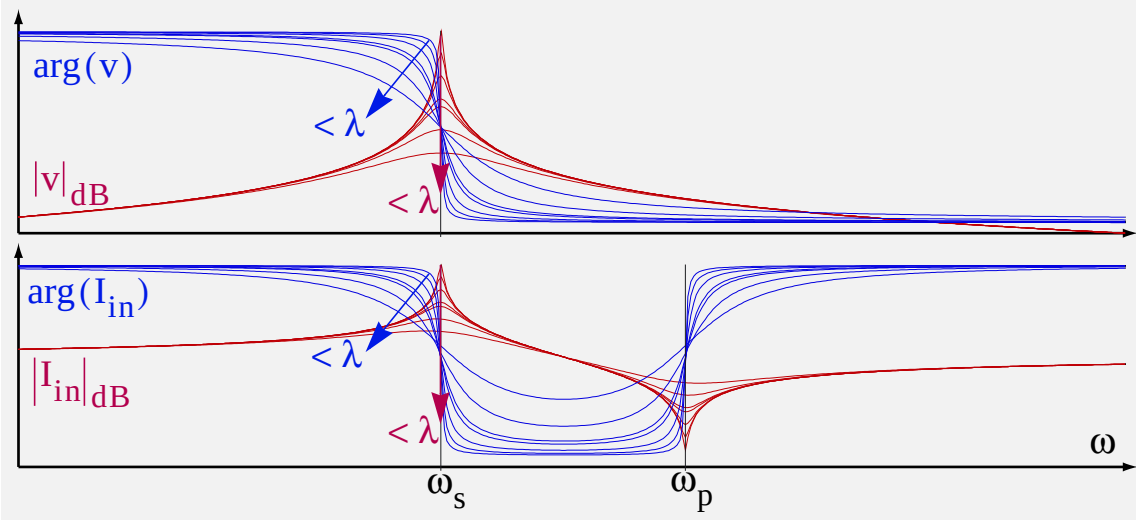


Figure 4-10. Simulations pour lesquelles un générateur de tension alternative est appliqué du côté électrique (pour différents amortissements λ). La partie mécanique est libre. Les graphes du haut présentent les diagrammes de Bode de la vitesse de la masse m. Les graphes du bas présentent les diagrammes de Bode du courant électrique.

L'élaboration de ce modèle nous a permis d'étudier le fonctionnement des éléments piézoélectriques et en particulier les trans-

ferts d'énergie entre le côté électrique et mécanique. Pour identifier les paramètres, nous aurions pu utiliser une méthode moins empirique (cf. "The System Identification Toolbox" de Mathworks). La méthode adoptée nous a cependant permis de bien comprendre les effets des paramètres sur les réponses du système. En particulier l'importance de la résistance de contact.

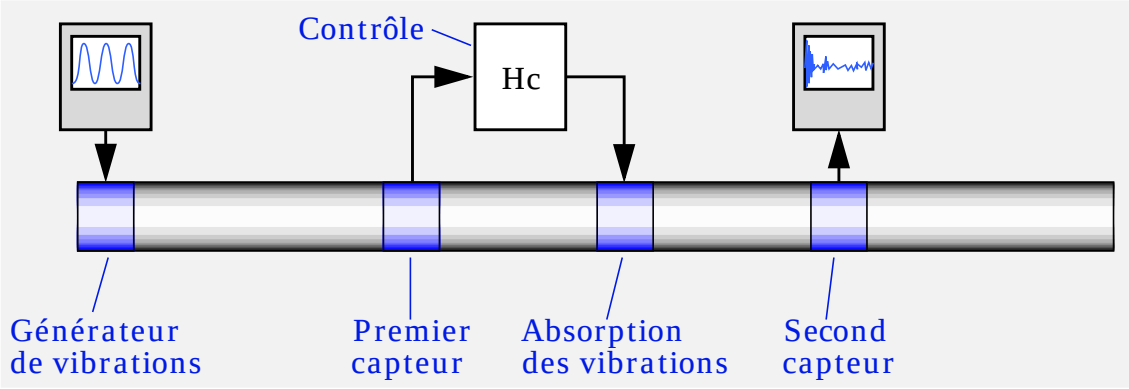


Figure 4-11. Dispositif permettant de générer des vibrations, de les mesurer, de les absorber et finalement de contrôler le résultat.

Ce modèle a été utilisé dans deux projets:

1. Dans le premier, nous nous sommes intéressés à l'amortissement actif de vibrations [Boe96a] [Boe96b] [Dau96] [Waa97]. Le dispositif étudié est composé de capteurs et d'actionneurs piézoélectriques permettant de générer, de mesurer et d'amortir grâce à une électronique de contrôle, des vibrations dans un barreau d'aluminium (fig. 4-11). Le second capteur permet de constater l'amortissement des vibrations. Selon notre modélisation, les vibrations sont entièrement absorbées. Lors des premières simulations, nous avons donc été surpris d'obtenir un signal non-nul à la sortie du second capteur. Ce signal ressemble à un bruit blanc de faible amplitude. Nous pouvons nous rendre compte qu'il est dû à la précision de la simulation car il peut être réduit ou augmenté en jouant avec certains paramètres du simulateur. Ce phénomène que nous avons appelé *bruit de simulation*, révèle quelque chose d'important:

Comme nous l'avons déjà mentionné, les simulateurs de type SPICE ont été conçus pour simuler des circuits électroniques. Ce type de simulateur résout chaque signal avec une certaine précision. Or cette précision est la même pour tous les signaux. C'est compréhensible lorsqu'il s'agit d'un circuit électronique. Mais, pour une microstructure, cela peut être gênant. Il se peut en effet que l'on ait besoin d'une grande précision pour un signal alors que l'on pourrait se contenter d'une faible précision pour un autre. Les simulateurs de type SPICE ainsi que le langage HDL-A ne permettent pas de contrôler la précision de signaux particuliers. En revanche le langage VHDL-AMS permet un contrôle de la précision des signaux. Actuellement il n'existe pas de simulateurs acceptant entièrement VHDL-AMS et en particulier la possibilité de contrôler la précision des signaux.

2. Dans le second projet (c.f. § 4.3.4), il était question d'un micromoteur ultrasonique [Rac93]. Le rotor du micromoteur (fig. 4-12) dont le design a été spécialement étudié, tourne grâce à des ultrasons émis par un stator piézoélectrique.

Nous retiendrons de la confection et de l'utilisation de ce modèle les points suivants:

- lorsque l'on doit coder des équations dans un modèle, il est plus facile de le faire en utilisant un HDL analogique que de créer un macro-modèle;
- généralement, les temps de simulation d'un macro-modèle sont plus courts que ceux d'un modèle utilisant un HDL;
- lorsque l'on modélise un sous-système en se basant sur l'énergie, l'approche macro-modèle est bien adaptée car il s'agit de réseaux de Kirchhoff (systèmes conservatifs);
- de nombreuses simulations en modifiant les paramètres permettent de mieux comprendre leur influence;
- la validation du modèle de l'élément piézoélectrique était également une opération délicate et importante;
- le simulateur génère un bruit qu'il faut être en mesure d'identifier;

- le contrôle séparé de la précision de simulation de certains signaux peut être important pour certains modèles.

4.3.4 Rotor d'un micromoteur ultrasonique

Un micromoteur ultrasonique est formé d'un rotor micro-usiné et d'un stator piézoélectrique (fig. 4-12). Le stator piézoélectrique émet des ultrasons verticalement en direction du rotor. La conversion des vibrations en mouvement rotatif est réalisée grâce à des pattes fixées au rotor. A l'état de repos, grâce à une précontrainte appliquée au rotor les pattes du rotor sont en contact avec le stator.

Les vibrations provenant du stator induisent des contraintes mécaniques. Pour comprendre le fonctionnement des pattes, nous pouvons décomposer le mouvement de la manière suivante (fig. 4-12 d & e).

1. Le rotor est en position de repos. La précontrainte le retient contre le stator.
2. La vibration provenant du stator écrase le rotor. Sous cette contrainte les pattes du rotor se déforment.
3. Le rotor retourne à sa position de départ.

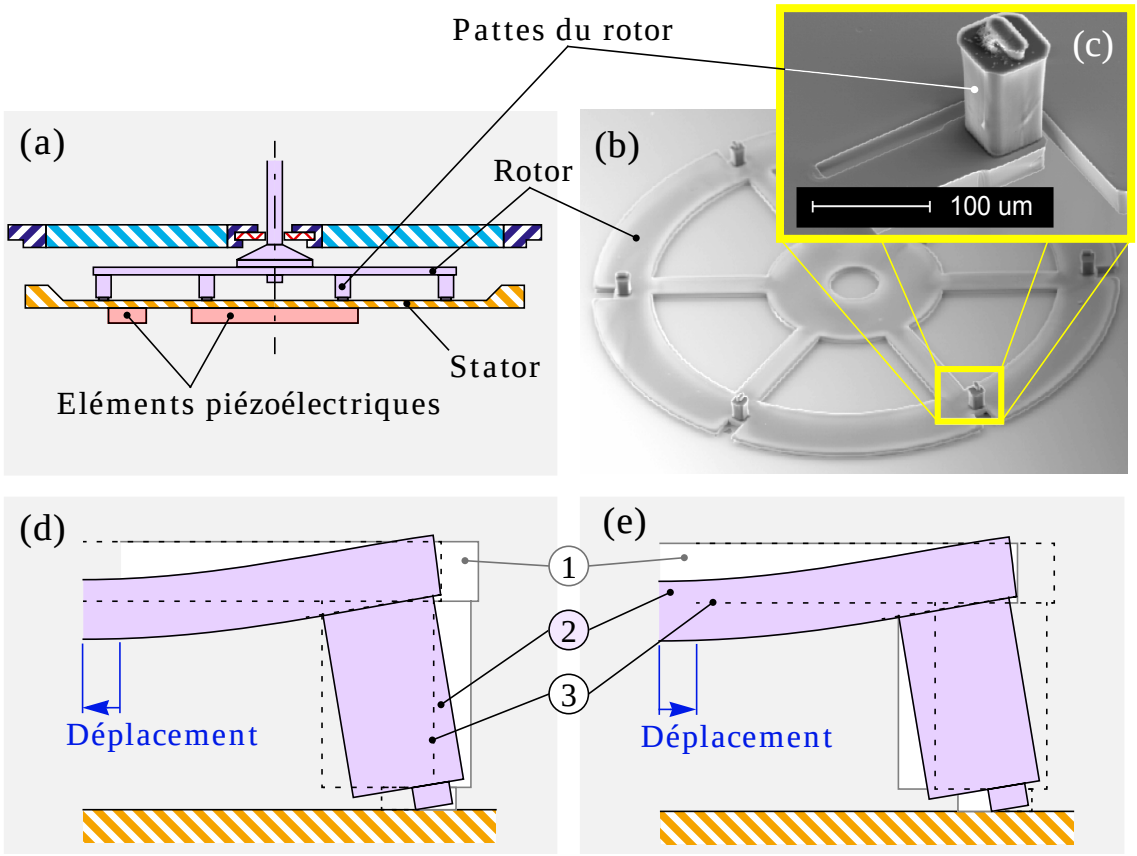


Figure 4-12. a) Coupe d'un micromoteur ultrasonique. (b) Photo d'un rotor micro-usiné. (c) Photo d'une patte du rotor. (d) & (e) Fonctionnement des pattes, décompositions du mouvement

Pour que le rotor se mette à tourner, il faut qu'il y ait glissement des pattes sur le rotor. Imaginons que l'on appuie doucement sur le rotor (fig. 4-12 d). Les pattes se déforment sans glisser sur le stator. Ensuite, le rotor est relâché rapidement. C'est en remontant que le glissement de la patte a lieu et que le rotor se déplace.

Imaginons maintenant que l'on appuie promptement sur le rotor (fig. 4-12 e). Les pattes du rotor se déforment tout en glissant sur le stator. Si l'on relâche doucement le rotor, les pattes ne glissent pas et le rotor se déplace dans la direction opposée au premier cas de figure. Dans quel sens notre dispositif tournera-t-il?

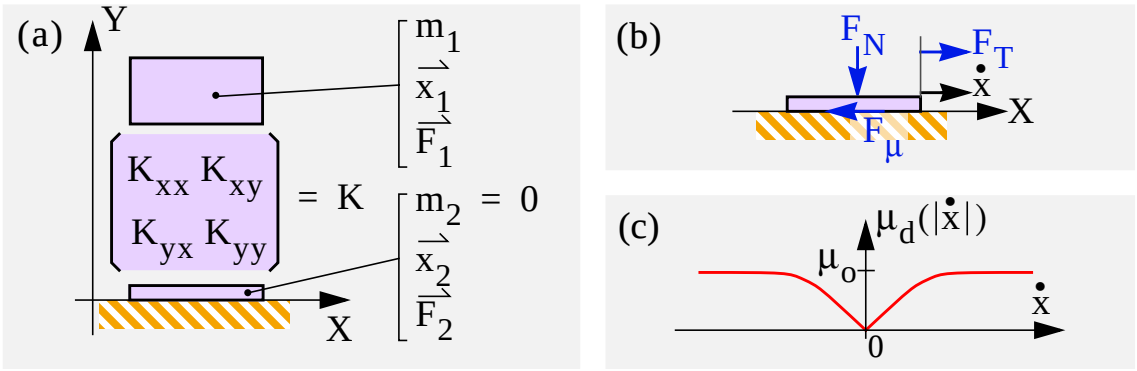


Figure 4-13. (a) Modélisation du rotor. (b) Frottement, dispositif considéré. (c) Graphique du coefficient de frottement visqueux en fonction de la vitesse.

Pour modéliser le rotor, nous le décomposons en trois parties (fig. 4-13 a): une masse (m_1) connectée par une matrice de rigidités (K) à une plaque sans masse (m_2=0) en contact avec une surface de référence. La masse m_1 représente la masse du rotor. Il s'agit d'une masse ponctuelle respectant la loi de Newton:

$$m_1 \cdot \ddot{\vec{x}}_1 = \sum \vec{F} \qquad (4.19)$$

La matrice des rigidités modélise [Del95] la transmission des forces entre le stator et le rotor:

$$\vec{F} = K \cdot \vec{x} \qquad (4.20)$$

où K est la matrice des constantes de ressort, $\vec{F}$ la force appliquée et $\vec{x}$ le déplacement.

Pour modéliser ce qui se passe entre les pattes du rotor et le stator, nous avons placé une plaque sans masse au bout de la matrice de rigidités. Le frottement entre la plaque et le stator (ou la surface de référence, fig. 4-13 b) est modélisé par un *frottement visqueux* lorsque la plaque est en mouvement et par un *frottement statique* lorsque la plaque n'est pas en mouvement. C'est-à-dire qu'il faut différencier le cas où la plaque est en mouvement du cas où la plaque est arrêtée (vitesse nulle). Lorsque la vitesse est non-nulle ($\dot{x} \cong 0$), le coefficient de viscosité dépend de la valeur absolue de la vitesse absolue (fig. 4-13 c):

$$\mu_d(|\dot{x}|) = \mu_o - \mu_o \cdot e^{-\frac{|\dot{x}|}{V_{th}}} \qquad (4.21)$$

où V_{th} est une vitesse de seuil et μ_o un coefficient de friction. La force de frottement est alors donnée par la relation suivante:

$$F_\mu(\dot{x}) = -\mathrm{sgn}(\dot{x}) \cdot \mu_d(|\dot{x}|) \cdot F_N \qquad (4.22)$$

Si la vitesse est nulle, nous avons les relations suivantes:

$$F_\mu = \begin{cases} -F_T & \text{si } |F_T| \le \mu_s \cdot F_N \\ -\mathrm{sgn}(F_T) \cdot \mu_s \cdot F_N & \text{si } |F_T| > \mu_s \cdot F_N \end{cases} \qquad (4.23)$$

où μ_s est le coefficient de frottement statique.

Modéliser ce qui se passe entre le rotor et le stator avec une plaque sans masse est une approximation de la réalité. Cette approximation introduit des discontinuités. En effet, au moment où l'on passe du frottement statique au frottement visqueux, nous avons une discontinuité de la position de la plaque. Il faut donc que le simulateur détecte quand la plaque passe du frottement statique au frottement visqueux et inversement.

Pratiquement, avec cette modélisation, nous avons constaté que lorsque la plaque est en frottement visqueux, elle ne retourne plus en frottement statique. Cela est également dû à un bruit de simulation. Pour que la plaque retourne à un frottement statique, il faut donc définir et détecter un seuil et forcer la plaque à retourner dans le régime de frottement statique.

Dans ce modèle, il faut également contrôler si la plaque est en contact ou non avec la surface de référence. Si la plaque n'est plus en contact avec la surface, il n'y a plus de frottement et donc plus de dissipation d'énergie selon l'axe X.

Pour coder ce modèle, nous avons utilisé HDL-A. La vue externe et donc le symbole de ce modèle (fig. 4-14 a) possèdent cinq bornes. Deux sont des bornes mécaniques caractérisées par deux grandeurs: vitesse et force. Ces deux bornes utilisent l'analogie

force-courant (FI). Les trois autres bornes sont des sorties caractérisées par une seule grandeur et donnent des informations concernant la plaque sans masse:

- la première indique le type de frottement (fr=0: signifie frottement statique et fr=1: signifie frottement visqueux),
- la seconde indique si la plaque est en contact ou non avec la surface de référence (ap=0: la plaque est en contact et ap=1: la plaque n'est pas en contact),
- la troisième donne la position de la plaque (x_2).

Remarquons qu'avec les modèles possédant des régimes de fonctionnement distincts, tels que: frottement statique, frottement visqueux, etc., il n'est pas possible d'effectuer des simulations AC. Notre modèle ne peut donc être utilisé que pour des simulations de type DC ou transitoire.

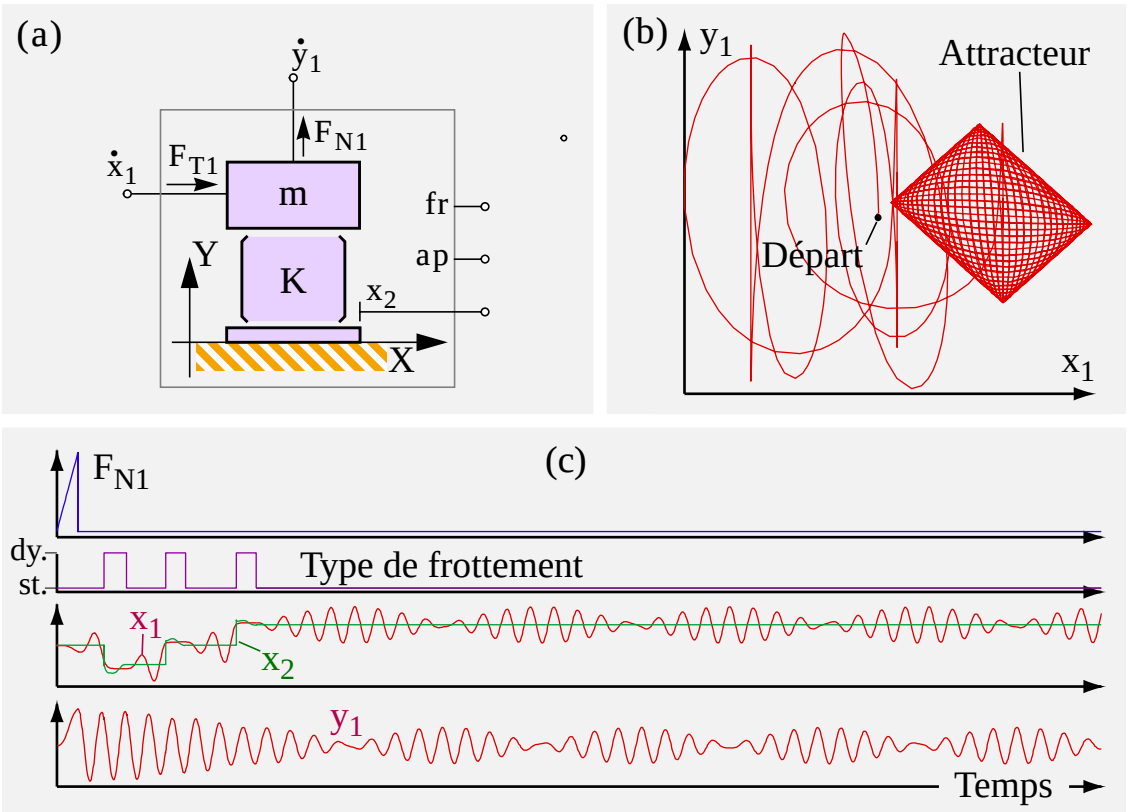


Figure 4-14. (a) Symbole du modèle du rotor. Résultat d'une simulation transitoire, (b) position de la masse du rotor, (c) graphiques avec en abscisse le temps et en ordonnée (de haut en bas): la force appliquée, le type de frottement, les positions selon X de la masse et de la plaque sans masse, la position de la masse selon Y.

Les figures 4-14 b & c montrent un exemple de simulation transitoire. Pour cette simulation une force vers le haut (F_{N1}) est progressivement appliquée à la masse du rotor. Après avoir atteint une certaine valeur celle-ci est annulée. Sur les graphiques temporels (fig. 4-14 c), nous constatons que la position (x_2) de la plaque sans masse présente des discontinuités. Nous pouvons également constater que cette plaque passe par des moments de frottement statique et dynamique. Le mouvement de la masse (fig. 4-14 b) est alors stochastique. Lorsque l'énergie de la masse n'est plus suffisante pour passer en frottement visqueux, il n'y a plus de dissipation d'énergie et la masse tombe dans un attracteur.

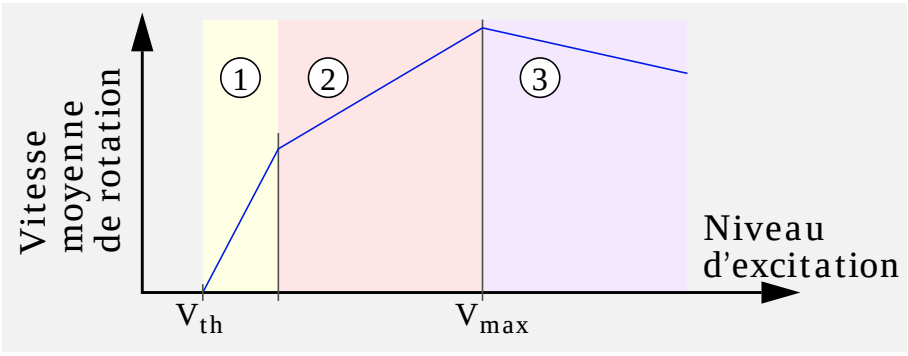


Figure 4-15. Esquisse du graphique représentant la vitesse moyenne de rotation du rotor en fonction du niveau d'excitation du micromoteur ultrasonique. On distingue trois modes de fonctionnements: (1) frottement unidirectionnel, (2) frottement bidirectionnel, (3) contact intermittent.

Ce modèle du rotor couplé au modèle de l'élément piézoélectrique a permis d'étudier l'influence du niveau d'excitation sur la vitesse moyenne de rotation du micromoteur ultrasonique (fig. 4-15). Différents modes de fonctionnements ont ainsi pu être mis en évidence:

1. Le rotor tourne grâce au frottement unidirectionnel présenté à la figure 4.12 d.
2. Le frottement est bidirectionnel alors que le rotor tourne dans le même sens que le premier mode de fonctionnement.
3. Le contact avec le rotor est intermittent.

Les résultats des simulations ont donc permis de déterminer le niveau d'excitation auquel la vitesse moyenne du rotor est la plus

grande. Il s'agit du point de passage entre le deuxième et le troisième mode de fonctionnement.

Nous retiendrons de l'élaboration et de l'utilisation de ce modèle les points suivants:

- Un modèle de type "larges signaux" utilisant différents modes de fonctionnements, c'est-à-dire un modèle composé de "IF", ne peut pas être utilisé pour des simulations de type AC.
- Certaines simplifications de la modélisation engendrent des discontinuités pouvant poser des problèmes au simulateur. La simplification d'un modèle n'est donc pas une opération triviale.
- Le bruit de simulation peut engendrer de grandes imprécisions. Pour éviter ces imprécisions, il faut compliquer le modèle.
- La modélisation et la simulation de microstructures, en particulier si l'on utilise l'approche analytique, peuvent être d'une aide précieuse à l'optimisation de celles-ci.

4.4 Exemple d'un banc de tests

Les procédures de tests d'un microsystème sont extrêmement importantes. C'est pourquoi, lorsqu'un microsystème est réalisé, il est soumis à des bancs de tests. Ces bancs de tests peuvent servir au calibrage, à tester des fonctionnalités, etc. Grâce à la simulation globale, il est possible d'étudier et de développer des modèles de bancs de tests avant qu'un microsystème n'existe. Ces bancs de tests peuvent ensuite servir tout au long du développement et même après, lors de la production.

L'exemple que nous présentons ici est un modèle de banc de tests relié à un microsystème. Le microsystème (c.f. § 5.1) permet de mesurer la température et la pression. Il est composé (fig. 4-16 b) d'un capteur de pression piézo-résistif, d'une interface d'électronique analogique, de convertisseurs A/N, d'une implémentation d'algorithmes de linéarisation sous la forme d'électronique numé-

rique et éventuellement de SW. Le résultat des mesures traitées par le microsystème est soit affiché, soit communiqué à un autre système. Le but du banc de tests est de mesurer la surface (cf. fig. 4-7 c) caractérisant la réponse du capteur de pression relié à l'électronique analogique.

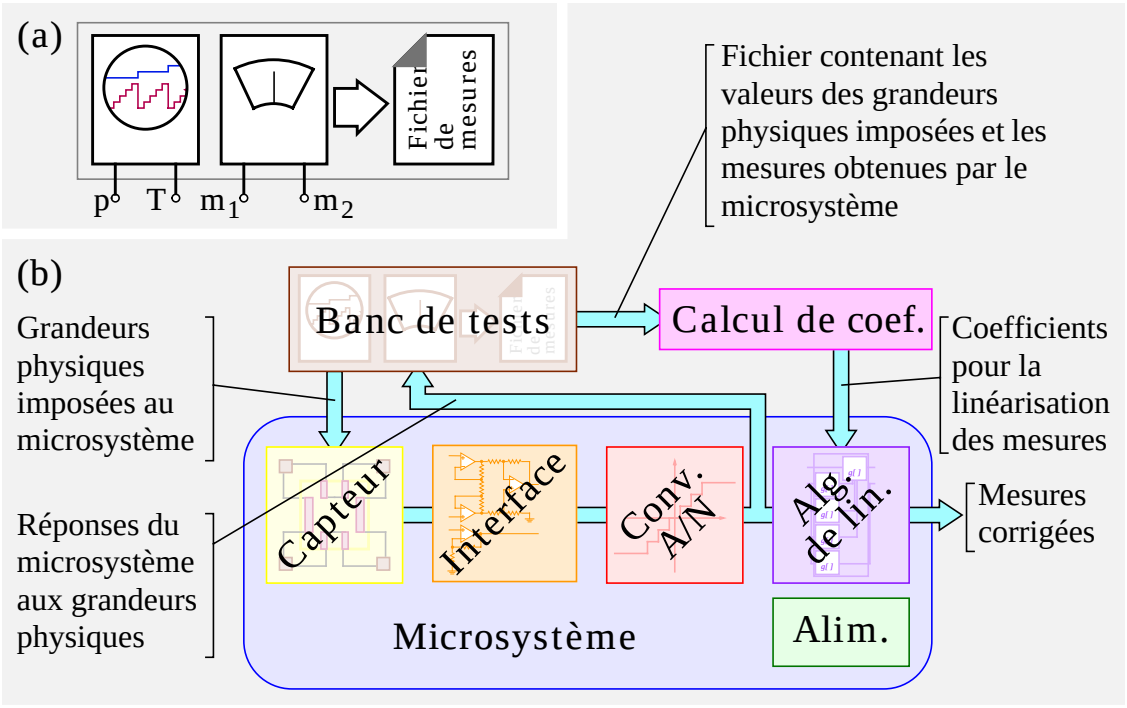


Figure 4-16. (a) Symbole du modèle du banc de tests. (b) Schéma de l'utilisation du banc de tests connecté à un microsystème.

Pour coder ce modèle, nous avons utilisé le langage HDL-A faisant appel à des fonctions développées en C. Le symbole de ce modèle (fig. 4-16 a) possède quatre bornes. Deux sont des sorties caractérisées par une grandeur; dans un cas il s'agit de la pression (p) et dans l'autre de la température (T). Les deux autres bornes sont des entrées (m_1 & m_2) permettant l'acquisition de mesures.

Ce modèle est destiné à être utilisé en simulation de type transitoire de durée déterminée. Son fonctionnement est le suivant (fig. 4-16 b): aux deux sorties (p & T) sont successivement appliquées des températures et pressions différentes. Après stabilisation de celles-ci, des mesures sont effectuées par les deux bornes (m_1 & m_2). Le résultat d'une simulation est un fichier contenant les températures et pressions appliquées ainsi que les mesures effectuées. Ce fichier est destiné à être traité afin d'en extraire des

coefficients. Les coefficients serviront aux algorithmes de linéarisation permettant, à partir des mesures provenant du capteur, de calculer la pression et la température appliquée.

Le modèle de ce banc de tests nous a permis d'étudier l'influence du nombre de mesures sur la qualité du calibrage d'un tel microsystème. Les procédures de calibrage en laboratoire étant coûteuses, cette approche autorise l'optimisation de celles-ci. Ce banc de tests nous a également permis d'étudier différentes manières de calculer les coefficients. Ainsi, très tôt dans le développement du microsystème, et même avant que celui-ci n'existe, une procédure de calcul des coefficients efficace a pu être proposée.

Ce modèle n'est pas un test fonctionnel du microsystème. Pour tester la fonctionnalité, il faut comparer les mesures corrigées aux grandeurs physiques imposées. La différence entre ces deux grandeurs représentant l'erreur. Nous le verrons plus loin (§ 5.1), c'est l'approche que nous avons adoptée avec succès dans le cadre d'un projet visant à réaliser un cicuit intégré spécifique (*ASIC*: Application-Specific Integrated Circuit) de l'électronique du microsystème décrit plus haut (fig. 4-16 b).

4.5 Exemples de convertisseurs et quantificateurs

Le passage du monde analogique au monde numérique se fait grâce à un convertisseur A/N. Dans les microsystèmes cette opération est cruciale et, comme nous l'avons déjà mentionné, non-linéaire. Le retour au monde analogique s'effectue grâce à un convertisseur N/A.

Considérons la vue externe du modèle d'un convertisseur A/N présenté à la figure 4-17 a. Ce symbole possède 4 bornes. Deux sont des bornes électriques caractérisées par deux grandeurs (a_1 & a_2). Les deux autres sont des bornes numériques dont l'une est une entrée (ck) et l'autre une sortie (d_1). L'entrée (ck) permet de cadencer le convertisseur grâce à un signal d'horloge. C'est à la sortie (d_1) qu'apparaît le résultat d'une conversion.

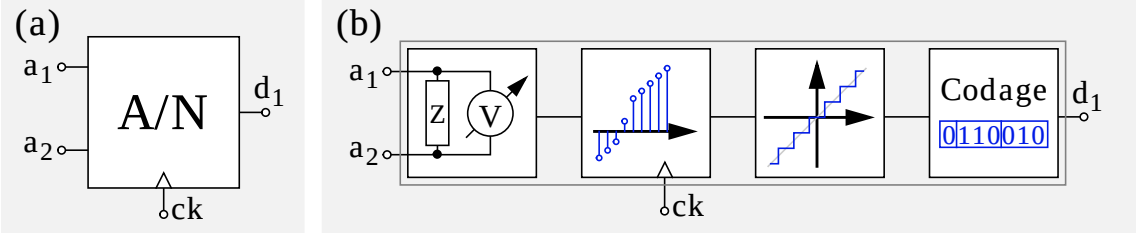


Figure 4-17. (a) Symblole d'un convertisseur A/N. (b) Vue interne comportementale d'un convertisseur A/N sous la forme d'une décomposition en quatre blocs.

La vue interne d'un convertisseur A/N peut être décomposée (forme comportementale) en quatre blocs (figure 4-17 b):

- un étage d'entrée composé d'une impédance dont la tension est mesurée par un voltmètre,
- un échantillonneur avec ou sans élément de maintien de la tension mesurée,
- un quantificateur
- et un bloc chargé de la transformation des échantillons en une séquence binaire.

La figure 4-18 montre différents quantificateurs Q(x) avec la fonction de distribution P(Er) de l'erreur correspondante. L'erreur étant définie de la manière suivante:

$$Er(x) = Q(x) - x \qquad (4.24)$$

Il existe différentes méthodes pour représenter un échantillon en une séquence binaire. Il est possible de les classer en deux familles: représentations à virgule fixe et représentations à virgule flottante. Dans la famille des représentations à virgule fixe, il y a la représentation: binaire naturelle, avec signe et amplitude, en complément à un, en complément à deux, etc. Chacune de ces représentations possède des avantages et des inconvénients dont l'étude sort du cadre de ce travail. Ce que nous pouvons retenir, c'est que pour cette famille:

- les nombres binaires représentent des nombres entiers et/ou fractionnaires (fig. 4-19 a),

- la résolution (Δ) est fixe sur la gamme.

Les représentations à virgule flottante binaire d'un nombre réel N sont généralement constituées (fig. 4-19 b) d'une mantisse M et de son signe multipliant une exponentielle de base deux dont l'exposant est P. La norme IEEE754 est consacrée à une représentation de ce type. Dans cette norme, les nombres de simple précision utilisent 32 bits: 1 bit pour le signe de la mantisse, 23 bits pour la mantisse et 8 bits pour l'exposant. Contrairement à la représentation en virgule fixe, la résolution (Δ) d'une représentation en virgule flottante est variable.

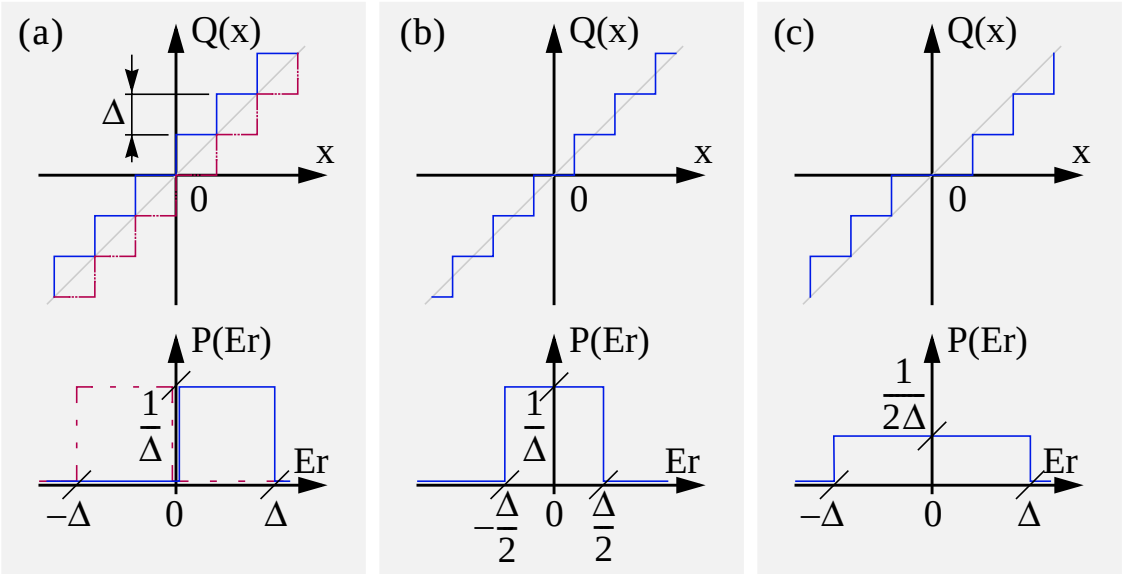


Figure 4-18. Différents types de quantificateurs avec, en bas, les fonctions de distribution de l'erreur. (a) Quantifications vers plus ou moins l'infini. (b) Quantification de type "arrondi" (*rounding*). (c) Quantification en direction de zéro.

Dans le cadre de l'approche MicroSIM, nous pouvons être amenés à utiliser un modèle de convertisseur plus ou moins réaliste. Il est par exemple envisageable d'éliminer la quantification dans le but d'étudier uniquement l'effet de l'échantillonnage sur le système. Inversement, il est possible de ne pas échantillonner le signal afin d'étudier l'effet de la quantification et de la représentation finie des nombres sur le système. Le convertisseur A/N se résume alors à un quantificateur sans délai. Pour encore simplifier le système, il est possible de ne pas fournir le résultat d'une conversion sous la forme d'un code binaire, mais sous la forme d'un nombre réel en base 10, plus facile à interpréter. Nous le verrons

plus loin (chapitre 5), cette approche permet de visualiser et d'étudier le bruit de quantification très tôt dans un projet.

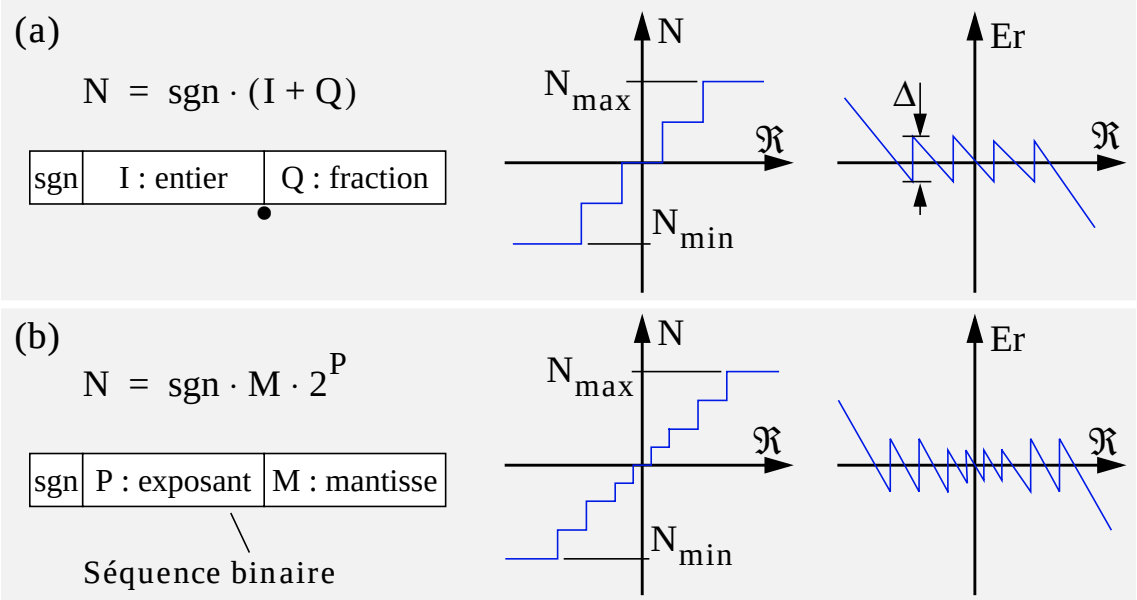


Figure 4-19. Représentation (a) à virgule fixe et (b) à virgule flottante. De gauche à droite: séquence binaire et signification mathématique, graphique des nombres pouvant être représentés (quantification), erreur de quantification et résolution.

Pour créer des modèles de quantificateurs utilisables avec un simulateur de type SPICE, nous avons utilisé HDL-A faisant appel à des fonctions développées en C. Il faut noter que ce type de modèles génère des discontinuités qu'un simulateur de type SPICE a parfois de la peine à simuler. Notons également que VHDL-AMS permet de créer les mêmes modèles sans faire appel à des fonctions C.

4.6 Exemples d'algorithmes

Dans le cas de l'amortissement de vibrations dans un barreau (fig. 4-11), il s'agissait de développer une électronique de contrôle (Hc). Lorsqu'il s'agit d'un capteur tel qu'un capteur de température (§ 4.3.1) ou un capteur de pression (§ 4.3.2), nous sommes confrontés au problème de la compensation et de la linéarisation des mesures. Comment peut-on résoudre ces problèmes de développement et d'implémentation d'algorithmes pour microsystèmes?

Pour les cas auxquels nous avons été confrontés, nous avons abordé le problème depuis le monde analogique.

Comme premier exemple, considérons le cas des vibrations dans un barreau (fonction de transfert Hc fig. 4-11). Pour l'étude et le développement de l'algorithme permettant d'amortir les vibrations (fonction de transfert Hc), il est possible de procéder de la manière suivante (fig. 4-20):

- Premièrement, différents algorithmes sont étudiés dans le domaine analogique sans se soucier des problèmes inhérents au monde numérique.
- Ce n'est que lorsque la compréhension du problème est suffisante et qu'un algorithme est choisi, que nous pouvons basculer dans le monde numérique en utilisant une transformation (par exemple (3.4) ou (3.5)).

Finalement, lorsque l'algorithme est simulé avec succès dans le monde numérique, une implémentation HW peut avoir lieu.

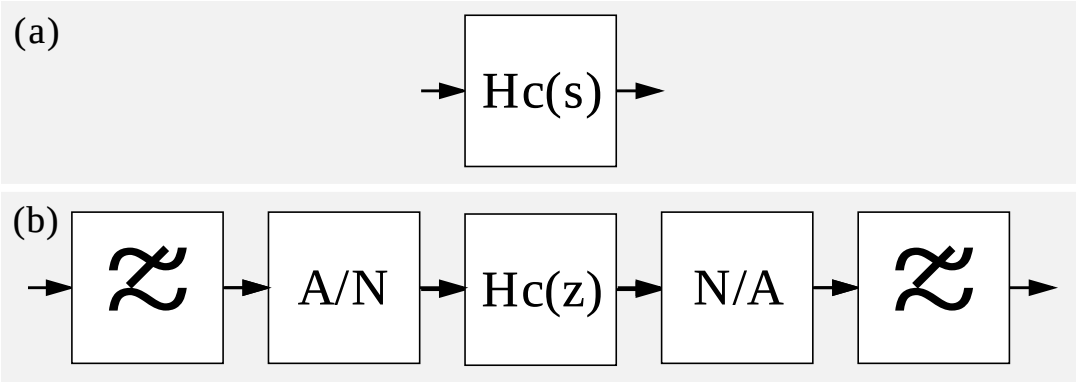


Figure 4-20. Développement d'un algorithme: (a) dans le monde analogique, (b) dans le monde numérique.

Comme second exemple, considérons le cas d'un capteur relié à une électronique analogique et numérique. L'électronique numérique est chargée du contrôle de l'électronique analogique et du capteur, ainsi que de la linéarisation des mesures.

Admettons que la réponse du capteur et de l'électronique analogique à un stimulus physique x soit la fonction f(x). Ce que l'électronique numérique reçoit (avant la conversion A/N) c'est donc f(x),

ou plus exactement des échantillons de f(x) notés $f_i=f(x_i)$. Le travail de l'électronique numérique consiste à extraire de cette information la grandeur du stimulus physique. L'approche choisie pour y parvenir est l'interpolation polynômiale [Ber96] [Hor98]. L'idée (fig. 4-21) est d'appliquer la relation fonctionnelle g[ ] à f(x) de manière à obtenir, sur un intervalle fermé [a,b], une approximation d'une fonction de x nommée fonction cible et notée q(x):

$$g[f(x)] = \sum_{i=0}^{n} u_i \cdot f(x)^i \cong q(x) \qquad x \in [a, b] \qquad (4.25)$$

où n est l'ordre et u_i les coefficients de la relation fonctionnelle. Généralement q(x) est un polynôme d'ordre un, mais cela pourrait être autre chose.

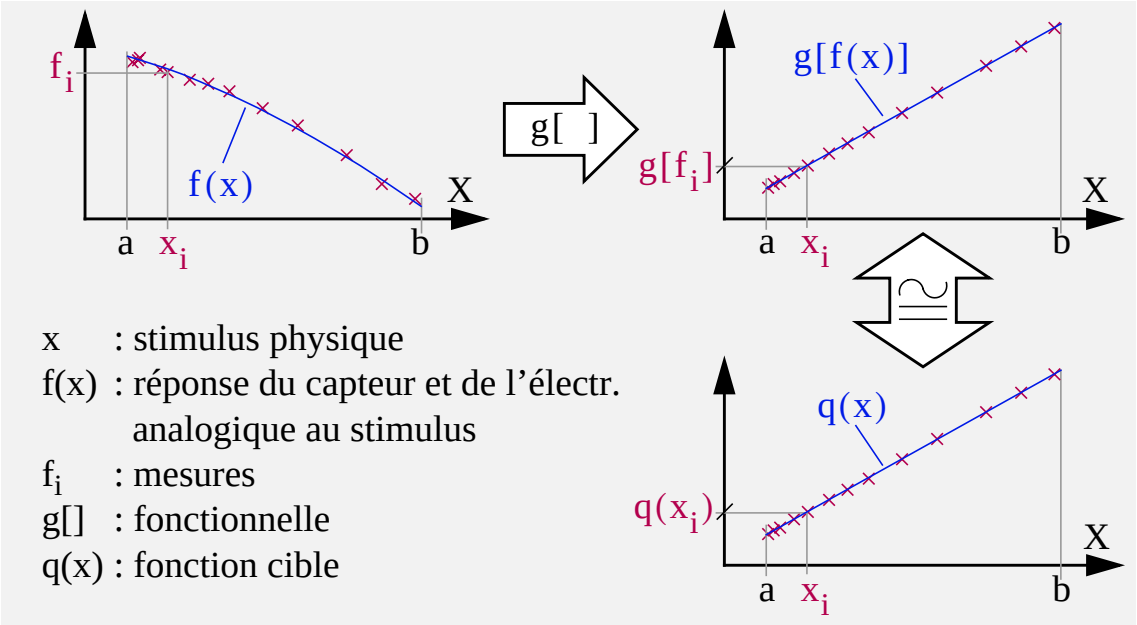


Figure 4-21. Principe de l'interpolation polynômiale.

Admettons que nous savons calculer les coefficients de la relation fonctionnelle à partir de la fonction cible et de la réponse du capteur et de l'électronique analogique. Nous pouvons constater que, pour implémenter l'algorithme, un certain nombre de choix sont à faire. Par exemple:

- choix de convertisseurs A/N,
- ordre n de la relation fonctionnelle g[ ],

- choix d'une représentation des nombres, etc.

Ces choix sont autant de degrés de liberté avec lesquels nous pouvons jouer pour optimiser l'implémentation. La simulation globale est un moyen permettant de comprendre l'influence de ces choix. Il est ainsi possible d'utiliser un simulateur analogique de type SPICE pour étudier l'influence de l'ordre n de la relation fonctionnelle g[ ].

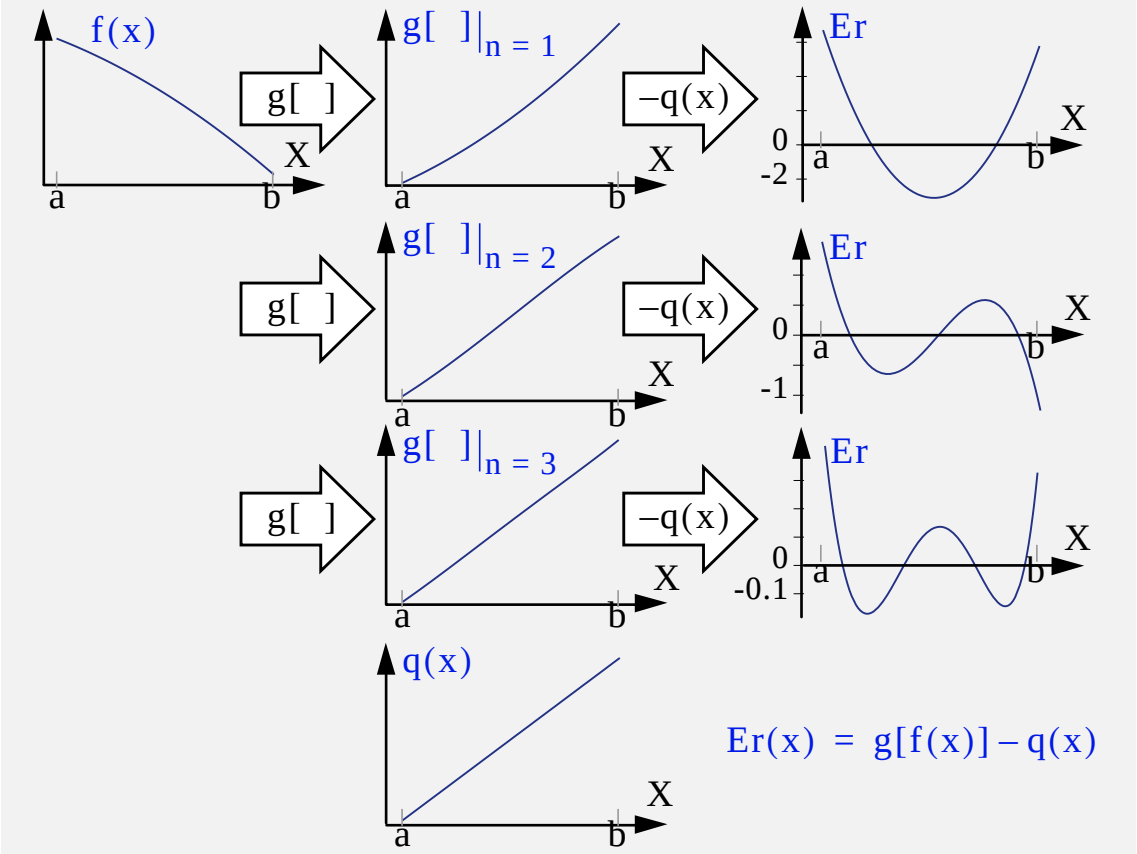


Figure 4-22. Etude de l'influence de l'ordre de la relation fonctionnelle g[ ] sur la qualité de la linéarisation.

La figure 4-22 présente les résultats de simulations avec un simulateur de type SPICE pour n=1, 2 et 3. La fonction d'entrée f(x) correspond à ce que l'on obtient si on utilise la résistance équivalente d'un capteur de pression piézo-résistif comme capteur de température. La fonction cible q(x) est un polynôme d'ordre 1. Comme nous pouvons nous y attendre, l'erreur Er(x) définie comme la différence entre la fonction obtenue g[f(x)] et la fonction cible q(x) diminue avec l'augmentation de l'ordre de la relation

fonctionnelle. Ce type de simulations permet d'évaluer cette diminution et par conséquent d'effectuer le choix de l'ordre.

Lorsque l'influence de l'ordre est correctement appréciée, on peut passer à l'implémentation de l'algorithme dans le domaine numérique. La prochaine étape consiste à choisir une représentation binaire des nombres ainsi que la longueur des mots binaires. Or, la longueur des mots binaires est liée au rapport signal sur bruit (*SNR*: Signal-to-Noise Ratio). Pour une distribution uniforme du signal, il est possible de démontrer que [Cou84] [Bri94]:

$$SNR_{dB} = 10 \cdot \log\left(\frac{P_S}{P_B}\right) \cong 6 \cdot N \qquad (4.26)$$

où P_S est la puissance du signal, P_B la puissance du bruit et N le nombre de bits utilisés par la dynamique du signal.

Il est également intéressant de noter que l'équation (4.25) peut être écrite de la manière suivante (schéma de Horner cf. § A.3.1):

$$g[f(x)] = ((u_n \cdot f(x) + u_{n-1}) \cdot f(x) + \ldots) \cdot f(x) + u_0 \qquad (4.27)$$

Or, si mathématiquement les équations (4.25) et (4.27) sont les mêmes, leur implémentation numérique ne donne pas forcément les mêmes résultats.

En effet, imaginons que l'on ait choisi une représentation des nombres à virgule fixe de 8 bits. Une multiplication de deux nombres donne un résultat de 16 bits. Si l'algorithme possède plusieurs multiplications, on conçoit qu'il faille tronquer les résultats après chaque opération. Ces opérations de troncation ont pour effet une injection de bruit. Or l'injection de bruit n'est pas la même pour les deux équations.

La figure 4-23 présente les résultats d'une simulation de l'implémentation de la relation fonctionnelle (4.27) avec un simulateur de type SPICE et pour une représentation des nombres de 8 bits, et une troncation après chaque opération. On peut constater que les bénéfices d'une relation fonctionnelle d'ordre 3 sont noyés

par le bruit. Pour diminuer ce bruit, il faudrait donc augmenter le nombre de bits de la représentation des nombres.

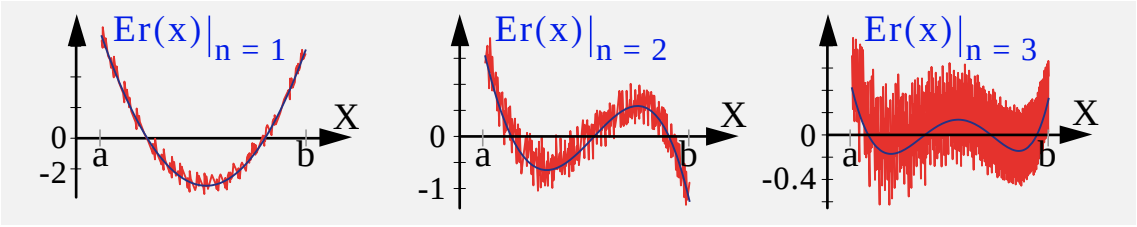


Figure 4-23. Simulations de l'erreur obtenue avec l'implémentation de la relation fonctionnelle (4.27) avec et sans représentation finie des nombres (8 bits) et troncation des résultats après chaque opération (schéma de Horner).

Nous savons que la dynamique du signal d'entrée f(x) doit être plus petite ou égale à la dynamique de la représentation des nombres. Ceci est également vrai pour les résultats partiels de la relation fonctionnelle g[ ]. Imaginons que l'on ait fixé l'ordre de celle-ci à trois. Nous avons donc cinq résultats partiels (4.28) devant se trouver à l'intérieur de la dynamique de la représentation des nombres.

$$g[f(x)]|_{n=3} = ((u_3 \cdot f(x) + u_2) \cdot f(x) + u_1) \cdot f(x) + u_0 \qquad (4.28)$$

m_1 s_1 m_2 s_2 m_2

La figure 4-24 présente les résultats d'une simulation de type SPICE du signal d'entrée, des résultats partiels et de la relation fonctionnelle d'ordre trois. Nous constatons que dans ce cas, la dynamique du signal d'entrée doit être beaucoup plus petite que la dynamique de la représentation des nombres (représentation signée à virgule fixe) pour que les résultats partiels n'induisent pas un débordement (Overflow). Un simulateur analogique de type SPICE peut donc également être utilisé pour étudier les phénomènes de débordement.

Ces deux exemples illustrent la voie que nous avons choisie de suivre. Nous cherchons à utiliser un simulateur analogique de type SPICE le plus longtemps possible. Les raisons de ce choix sont multiples. Les principales sont les suivantes:

- Premièrement ce type de simulateur permet d'effectuer des simulations de type AC. Ce que ne permet pas un simulateur numérique.
- Deuxièmement les microsystèmes sont toujours composés de parties purement analogiques, les capteurs et les actionneurs, constituant le point de départ du développement d'un micro-système.
- Troisièmement cette approche permet de n'utiliser qu'un seul simulateur le plus longtemps possible. Ce n'est qu'après avoir fixé le plus de paramètres que l'on passe à un simulateur mixte. L'implémentation de l'algorithme peut alors avoir lieu, soit en utilisant du HW uniquement, soit par du HW et du SW. Le problème de la séparation HW-SW (HW-SW Partitioning), que nous n'aborderons pas dans ce travail, est alors posé.

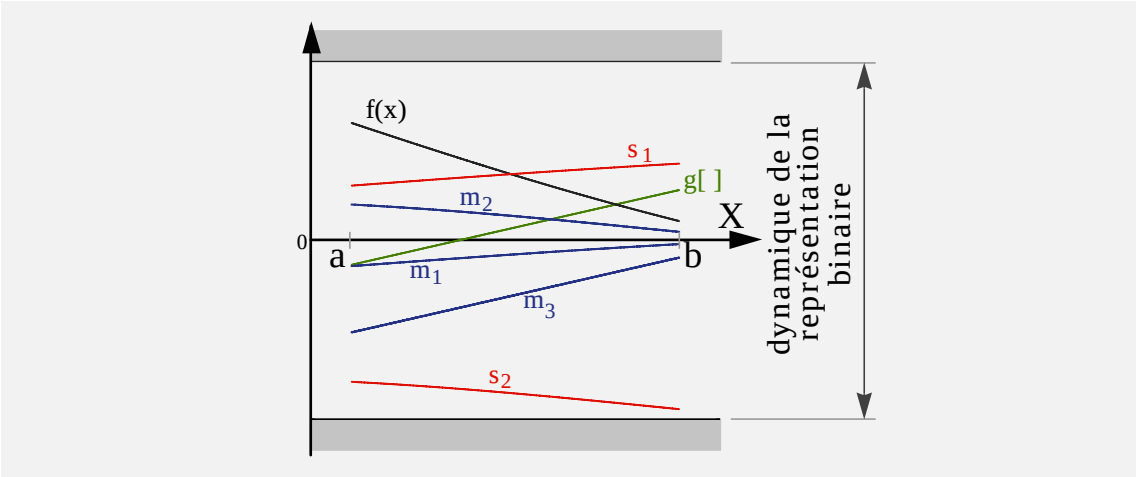


Figure 4-24. Simulation du signal d'entrée, des résultats partiels et de la relation fonctionnelle d'ordre trois, sans quantifications ni troncations.

4.7 Conclusion

Dans ce chapitre, nous avons présenté une adaptation du diagramme Y de Gajski et Kuhn pour les microsystèmes. Celui-ci permet d'énumérer, avec différents niveaux d'abstraction, les formes comportementales, les composants structurels et les objets physiques d'un microsystème ou d'un sous-système. Ce dia-

gramme donne une idée des différents types de modèles dont le designer de microsystèmes a besoin pour suivre l'approche MicroSIM.

Nous avons également présenté notre "design-kit pour la simulation globale de microsystèmes". Il s'agit d'une extension pour un environnement de CAO du commerce. Cette extension permet de suivre l'approche MicroSIM lors de la conception d'un microsystème. Celle-ci contient le système de fenêtres et de modèles utiles à la co-simulation HW-SW. Elle contient également une bibliothèque de modèles tels que: microstructures, sources pour créer des bancs de tests, quantificateurs, convertisseurs A/N, etc.

Lors de l'élaboration de certains modèles de cette bibliothèque, des difficultés ont été éprouvées. Dans ce chapitre, nous avons présenté quelques exemples de modèles les illustrant. Nous avons par exemple pu constater que:

- même simple, un modèle peut poser des problèmes de convergence à un simulateur analogique;
- pour construire de "bons" modèles, il ne suffit pas de connaître le langage de modélisation, il faut également connaître le simulateur;
- un modèle purement analytique ne décrit pas forcément la réalité avec suffisamment de détails;
- l'opération visant à valider un modèle est fondamentale et délicate;
- le simulateur analogique génère un bruit qu'il faut être en mesure d'identifier;
- élaborer un modèle simplifié n'est pas une opération triviale.

Dans ce chapitre, un exemple de banc de tests a également été présenté. Il montre que l'approche MicroSIM et la simulation globale permettent de développer des bancs de test avant qu'un microsystème n'existe physiquement.

Pour étudier un algorithme avant son implantation, nous avons montré qu'il est possible d'utiliser un simulateur analogique

de type SPICE. L'avantage de ce choix est qu'il autorise l'exécution de simulation de type AC. De plus, les microsystèmes sont toujours composés de parties purement analogiques. Ce choix permet donc de n'utiliser qu'un seul simulateur au début du développement d'un microsystème. Ce n'est que plus tard, lorsque le plus de paramètres possible sont fixés que l'on passe à un simulateur mixte.

4.8 Références

[Asc82] G.Asch, *"Les capteurs en instrumentation industrielle"*, Ed. Dunod, Paris, 1982

[Ber96] C.Berthoud, M.Ansorge, F.Pellandini, *"Effective static response compensation suitable for low-power ASIC implementation with an application to pressure sensors"*, Proceedings IMTC'96, Jun. 1996, pp. 1168-1173

[Boe95] A.Boegli, *"Compensation de capteurs de pression piézo-résistifs"*, Rapport technique No 705, CSEM, 1995

[Boe96a] A. Boegli, V. Moser, H.-P. Amann, F. Pellandini, B. Romanowicz, Ph. Lerch, M. Laudon, Ph. Renaud, *"System-Level Simulation Using Mixed-Nature Models"*, 1st Europe-Asia Congress on Mechatronics, Besançon, France, Oct. 1-3, 1996, p.12-17

[Boe96b] A.Boegli, *"Amortissement d'une table de mesure, premier essai de modélisation et régulation"*, Rapport interne, IMT 405 PE 5/96

[Bri94] J. E. Brignell, N. M. White, *"Intelligent Sensor Systems"*, Institute of Physics Publishing, Bristol and Philadelphia, 1994

[Cou84] F.de Coulon, *"Théorie et traitement des signaux"*, Traité d'électricité, Volume XI, Presses polytechniques romandes, 1984

[Dau96] D.Daudet, *"Amortissement actif de vibrations"*, Rapport interne, IMT 411 PE 6/9

[Del95] L.Dellmann, *"Conception, réalisation et comparaison de différents rotors pour micromoteur ultrasonique à liaison élastique"*, Travail de diplôme, IMT, Université de Neuchâtel, 1995

[Gaj83] D.Gajski, R.Kuhn. New VLSI Tools - Guest Editors' Introduction. IEEE Computer 16(12): pp. 11-14, Dec., 1983

[Hor98] G. van der Horn, J. L. Huijsing, *"Integrated Smart Sensors, Design and Calibration"*, Kluwer Academic Publishers, 1998

[Ike96] T.Ikeda, *"Fundamentals of Piezoelectricity"*, Oxford University Press, 1996

[Jos96] T.Jost, *"Etude et réalisation d'un démonstrateur pour la compensation de la mesure de capteurs de pression piézorésistifs"*, Projet de semestre d'été 1996, Institut de Microtechnique, Université de Neuchâtel

[Men99] *"Products at Mentor Graphics"*, http://www.mentorg.com/

[Rac93] G.-A.Racine, R.Luthier, N.F. de Rooij, *"Hybrid ultrasonic micromachined motors"*, MEMS'99 Proceedings, pp.128-132, 1993

[Rom97c] B.Romanowicz, Y.Ansel, M.Laudon, C.Amacker, Ph.Renaud, A.Vachoux, G.Schröpfer, *"VHDL-1076.1 Modeling Examples for Microsystem Simulation"*, 2nd Workshop on Libraries, Component Modeling and Quality Assurance, accompanying Computer Hardware Description Languages Symposium 97, Toledo, Spain, April 1997

[Waa97] B.J.W.Waarsing, *"Active Vibration Damping: Modeling, simulation and control of a complete microsystem"*, Internal Report, IMT, University of Neuchâtel, Nov. 1997

[Wal85] R.A.Walker, D.E.Thomas, *"A Model of Design Representation and Synthesis"*, 22nd Design Automation Conference, pp.453-459, 1985

Chapitre 5

Exemples de réalisations

Dans les chapitres précédents, nous avons vu ce qu'est l'approche MicroSIM et nous avons évoqué les outils nécessaires à sa mise en oeuvre. Nous avons également constaté qu'actuellement le point faible de cette approche se situe au niveau des modèles de microstructures. En effet, peu de bibliothèques de modèles de microstructures existent alors qu'il n'est pas facile de créer de bons modèles. De plus, les simulateurs disponibles peinent parfois à manipuler de tels modèles. Le chapitre précédent présente un design-kit pour la simulation globale de microsystèmes contenant une bibliothèque de microstructures ainsi que des exemples de modèles de microstructures ayant posé des problèmes.

Ce chapitre présente deux exemples de réalisation de microsystèmes composés de capteurs et d'électronique, mais pas d'actionneurs. Pour ces deux exemples, l'électronique est chargée d'alimenter les capteurs, d'acquérir des mesures et de les traiter.

1. Le premier exemple est un microsystème permettant de mesurer la pression et la température. Il est composé d'un capteur de pression piézo-résistif relié à une électronique. Deux implantations de l'électronique seront présentées: la première est un ASIC alors que la seconde utilise des composants du marché. Nous verrons que pour réaliser ces implantations, nous avons suivi l'approche MicroSIM.

2. Le second exemple est un instrument de mesure de la température du pH et de la conductivité de liquides. La taille de ce microsystème est celle d'un gros stylo. Il est composé de deux parties, la pointe contenant les capteurs ainsi qu'une mémoire, et le corps contenant l'électronique ainsi qu'une source d'énergie. L'interface utilisateur de cet instrument est un petit écran et un bouton poussoir. Pour réaliser ce second exemple, nous n'avons pas suivi la seconde phase de l'approche MicroSIM, à savoir la modélisation post-design. Nous nous en expliquerons (c.f. § 5.2.3).

5.1 Premier exemple, mesure de la pression

Le premier exemple de conception et de réalisation que nous avons choisi de présenter est celui d'un capteur piézo-résistif relié à une électronique (fig. 4-16). Le capteur est, par exemple, un capteur de pression, alors que l'électronique est chargée:

- de l'alimentation du capteur,
- de l'acquisition de mesures,
- de la compensation en température des mesures,
- de la linéarisation des mesures,
- de la communication avec le monde extérieur.

Dans ce contexte, la compensation en température et la linéarisation des mesures sont des fonctionnalités que l'on peut qualifier d'intelligentes. A condition que ses dimensions soient suffisamment réduites, ce système correspond à notre définition d'un microsystème (cf. § 2.1).

Avant de commencer la conception d'un tel microsystème, que ce soit en suivant l'approche MicroSIM ou une autre, il faut disposer de spécifications. Celles-ci ont pour but de transmettre le plus d'informations possibles au(x) designer(s) de manière à former un cadre. Hormis le fait qu'il ne faut pas en omettre, une des difficultés est de distinguer celles qui sont absolument nécessaires de

celles qui ne le sont pas. Etablir les spécifications d'un microsystème est donc un travail délicat demandant de l'expérience.

Dans le cas de ce premier exemple, il nous a paru important que soient spécifiées:

- les températures et pressions de fonctionnement,
- la communication avec le monde extérieur,
- l'alimentation du microsystème,
- la dépendance en température des mesures compensées,
- les non-linéarités acceptables,
- les bandes passantes,
- la précision des mesures compensées et linéarisées.

Pour fixer certains degrés de liberté et/ou donner des directives au(x) designer(s), il est envisageable de préciser des points tels que:

- les températures et pressions de stockage,
- les chocs mécaniques tolérables,
- la résistance à un environnement chimiquement hostile,
- les surtensions électriques supportables,
- la consommation électrique,
- le prix de vente,
- le calibrage du microsystème,
- les procédures de tests,
- etc.

Nous le constatons, les spécifications du microsystème sont étroitement liées au but visé. Idéalement, il faudrait disposer de spécifications complètes et définitives avant de commencer la première étape de l'approche MicroSIM, à savoir la modélisation comportementale. Nous avons pu constater que la pratique est souvent très

éloignée de cet idéal. Il n'est en effet pas rare de débuter un développement avec peu de spécifications.

Pour la suite de la présentation de ce premier exemple, nous admettons connaître les spécifications du microsystème. Nous les préciserons au besoin.

5.1.1 Modélisation comportementale

Introduction.

Une des spécifications du microsystème est l'utilisation d'une famille de capteurs de pression piézo-résistifs du commerce. A partir de spécifications des fabricants, il est possible d'obtenir des paramètres pour le modèle comportemental présenté § 4.3.2.

A la figure 5-1 on peut voir le diagramme bloc d'une première partition du microsystème (sans alimentation). Celui-ci est divisé en quatre parties:

- capteur,
- interface analogique,
- conversion A/N,
- compensation/linéarisation.

A la sortie du microsystème, nous avons placé un bloc chargé de dénormaliser les mesures provenant du microsystème. Le but est d'obtenir des mesures ayant les mêmes unités que les grandeurs d'entrées de manière à pouvoir les comparer.

Notons que l'entrée pression (P) du microsystème agit uniquement sur le capteur, alors que l'entrée température (T) agit sur le microsystème entier. Le capteur est relié à une électronique analogique chargée d'une première mise en forme des mesures de la température et de la pression. Mathématiquement, cette opération peut être modélisée par une soustraction (offset) et une amplification (G). A la sortie de l'interface analogique, nous obtenons deux

tensions. La première est une fonction de la température (VT) alors que la seconde dépend de la pression et de la température (VP). Ces tensions sont appliquées aux entrées de convertisseurs A/N.

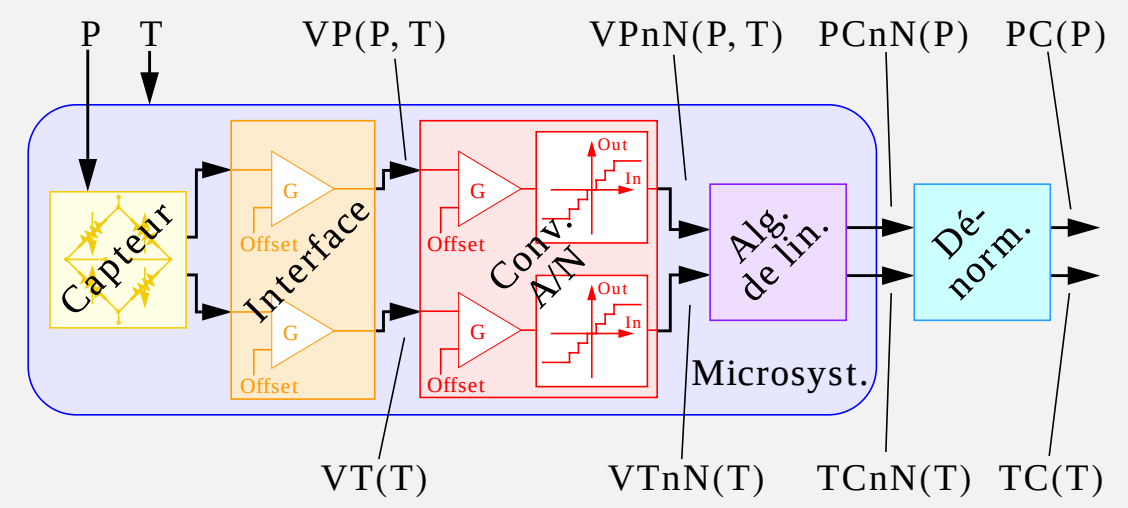


Figure 5-1. Diagramme bloc du microsystème (sans l'alimentation).

En première approximation, la conversion A/N peut être modélisée par une opération de normalisation (offset & G) suivie d'une quantification. Après la conversion, nous avons donc des tensions normalisées et numérisées représentant la température (VTnN) et la pression (VPnN). La conversion A/N nous fait basculer dans le monde numérique et un algorithme est alors chargé de:

- compenser en température les mesures de la pression,
- linéariser les mesures de la température et de la pression.

A la sortie du bloc nommé "algorithme de linéarisation", nous obtenons la température (TCnN) et la pression (PCnN) compensée, normalisée et numérisée. Pour simuler ce premier diagramme bloc, il faut encore étudier et choisir la méthode que nous voulons adopter pour compenser et linéariser les mesures.

Algorithme de compensation et de linéarisation

Pour compenser et linéariser les mesures provenant du capteur et traversant l'interface analogique et les convertisseurs A/N, nous

avons choisi d'adopter l'interpolation polynômiale évoquée au chapitre 4 (cf. § 4.6).

L'idée est d'implanter la relation fonctionnelle (5.1) de manière à former une unité de traitement polynômial (*PPU*: Polynomial Processing Unit).

$$g[f(x)] = ((u_n \cdot f(x) + u_{n-1}) \cdot f(x) + \ldots) \cdot f(x) + u_0 \qquad (5.1)$$

La figure 5-2 a, présente le symbole d'une PPU. Celle-ci possède 3 bornes:

- la grandeur à traiter (f),
- les coefficients de la relation fonctionnelle (u_i),
- le résultat du calcul de la relation fonctionnelle (g).

Toutes sont des bornes numériques, deux sont des entrées (f & u_i) alors que la dernière (g) est une sortie. Plus précisément, si la relation fonctionnelle est de degré n, l'entrée u_i est composée de n+1 entrées.

Pour évaluer la relation fonctionnelle de degré n, il suffit de multiplier et d'additionner un nombre fini de fois des nombres réels. Il n'y a donc pas de division à implanter. Remarquons que l'évaluation de l'équation (4.25) demande l'évaluation de n (n+1) multiplications et n additions. En revanche, le schéma de Horner (4.27) ne requiert que n multiplications et n additions. C'est donc bien celui-ci que nous désirons implanter.

Pour linéariser les mesures de la température, nous savons qu'une seule PPU suffit. Mais combien de PPU doit-on utiliser pour compenser et linéariser les mesures de la pression?

La linéarisation des mesures de la pression requiert une PPU dont les coefficients dépendent de la température. Il faut donc des PPU afin de calculer les coefficients de la première PPU. Deux dispositions sont envisageables (fig. 5-2 b & fig. 5-2 c). Ces deux dispositifs sont alimentés par des coefficients constants. Il s'agit des coefficients (CoTC) pour le calcul de la température linéarisée et des coefficients (CoCoPC) pour le calcul des coefficients (CoPC)

utiles au calcul de la pression compensée et linéarisée. Le premier dispositif utilise la mesure de la température (VTnN) pour calculer les coefficients (CoPC). Le second effectue en premier lieu la linéarisation de la température (TCnN). C'est cette dernière qui est utilisée pour le calcul des coefficients (CoPC).

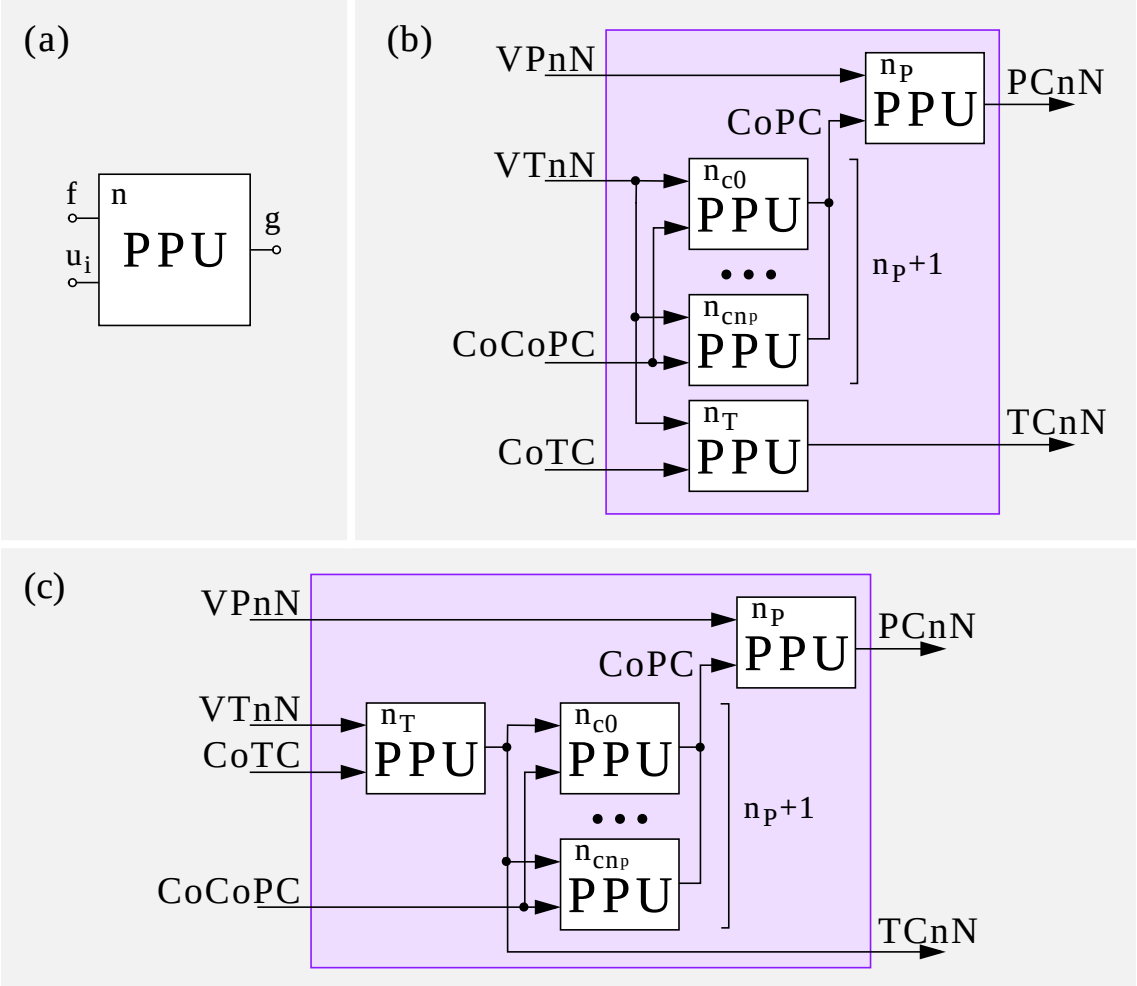


Figure 5-2. (a) Symbole d'une unité de traitement polynômiale. (b) & (c) Dispositifs autorisant le calcul des températures et pression compensée et linéarisée à partir de coefficients et de mesures.

La question que l'on se pose alors est: quel dispositif doit-on utiliser? Cette question en appelle d'autres:

- Comment calculer les coefficients (CoTC & CoCoPC) à partir de mesures?
- Comment obtenir des mesures pour calculer les coefficients?
- Quelles sont les mesures nécessaires et combien en faut-il?

- Quels convertisseurs A/N utiliser (dynamique d'entrée, précision, bande passante, etc.)?
- Dans le domaine numérique, quelle représentation des nombres utiliser (représentation en virgule fixe, en virgule flottante, en complément à un, en complément à deux, etc.)?
- Comment implanter les multiplications et additions des PPU (troncation des résultats)?
- Quel ordre (n_T, n_P, n_{c0}, n_{c1}, ..., n_{cnp}) des relations fonctionnelles g[] choisir?
- Comment alimenter le capteur (avec une tension ou un courant constant)?
- Comment implanter l'interface analogique?
- etc.

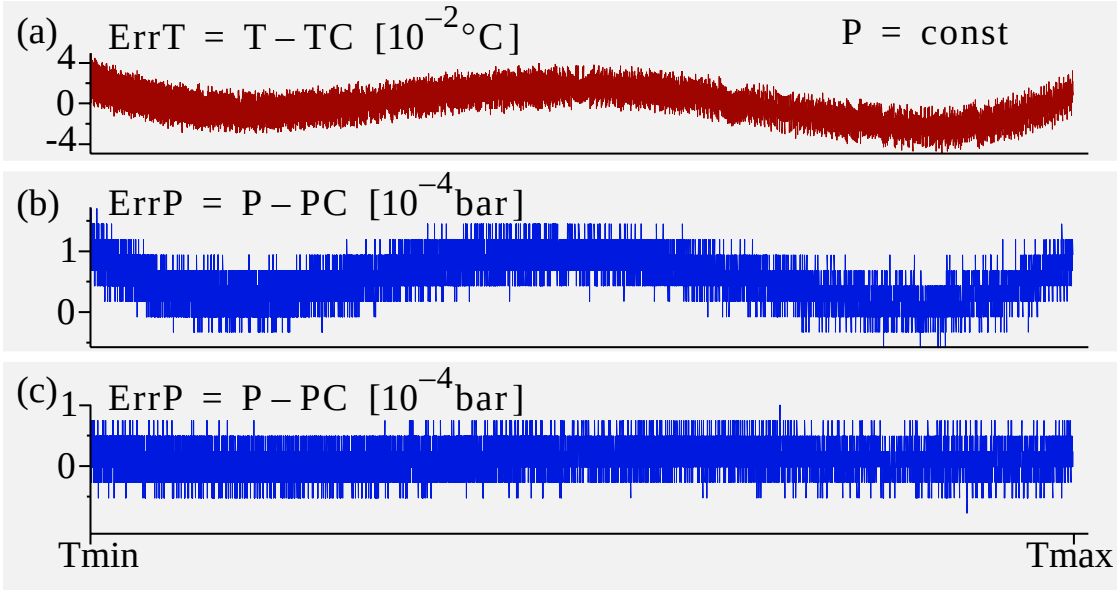


Figure 5-3. Comparaison des résultats obtenus avec les deux dispositifs de la figure 5-2. Pour ces simulations, $n_T=n_P=n_{c0}=n_{c1}$, ..., $n_{cnp}=3$. (a) Erreur de la linéarisation de la mesure de la température pour les deux dispositifs. Erreur de la compensation et de la linéarisation de la mesure de la pression, (b) dans le cas correspondant à la figure 5-2 b et (c) dans le cas correspondant à la figure 5-2 c.

Discussion.

Nous le constatons, beaucoup de questions surgissent. A ce stade, il ne faut pas perdre de vue les buts de la modélisation comportementale (cf. § 2.3.1) et empiéter sur la seconde phase de l'approche MicroSIM, à savoir la modélisation post-design.

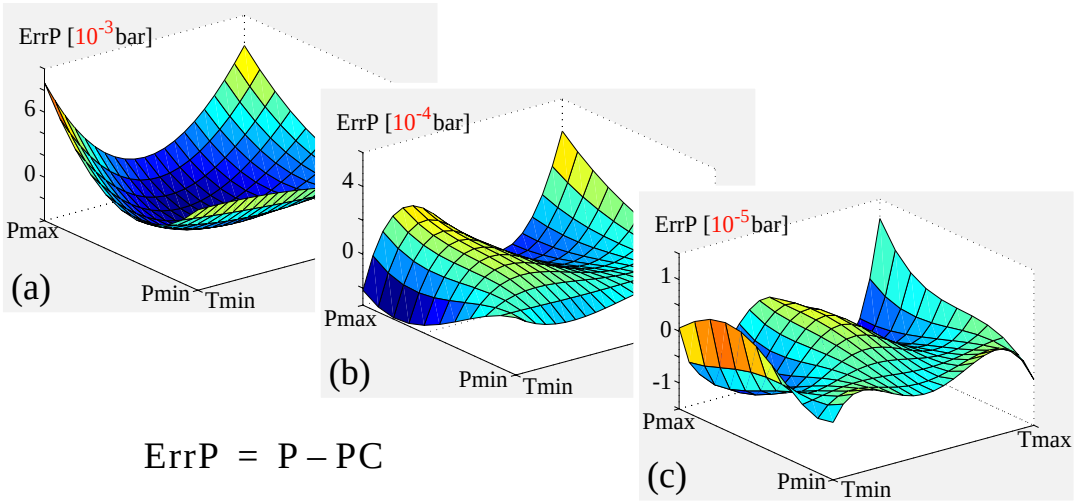


Figure 5-4. Influence de l'ordre des PPU sur la compensation et la linéarisation de la mesure de la pression. Graphiques obtenus avec le dispositif de la figure 5-2 b pour: (a) $n_T=n_P=n_{c0}=n_{c1}$, ..., $n_{cnp}=1$, (b) $n_T=n_P=n_{c0}=n_{c1}$, ..., $n_{cnp}=2$, (c) $n_T=n_P=n_{c0}=n_{c1}$, ..., $n_{cnp}=3$. Ces résultats sont obtenus sans quantification lors de la conversion A/N, ni représentation finie des nombres dans les PPU.

Pour cette première phase, nous avons développé des modèles comportementaux pour chacun des blocs de manière à répondre à certaines des questions posées. Nous avons par exemple pu étudier:

- les procédures de calibrage (cf. § 4.4) de manière à les minimiser;
- différentes façons de calculer les coefficients et choisir la plus adaptée (cf. § A.3.3);
- l'effet de l'ordre des relations fonctionnelles sur la compensation et la linéarisation de la mesure de température (fig. 4-22) et de la pression (fig. 5-4) afin d'effectuer un choix compatible avec les spécifications;

- l'effet de la représentation finie des nombres (fig. 4-23) et choisir celle permettant d'atteindre les spécifications;
- les différences (fig. 5-3), entre les deux dispositions des PPU présentées figure 5-2 b & figure 5-2 c.

Conformément aux buts de cette première étape, nous avons décomposé le microsystème en sous-système, et la simulation globale du microsystème nous a permis de valider cette décomposition. A la fin de cette première étape, nous disposons donc de modèles pouvant être utilisés comme spécification pour la deuxième phase: la modélisation post-design.

5.1.2 Modélisation post-design

Introduction

Les modèles des sous-systèmes développés lors de la première phase sont des modèles comportementaux qu'il s'agit d'implanter. La figure 5-5 présente les schémas blocs de deux implantations correspondant à des spécifications différentes. Dans les deux cas, il s'agit de schémas qu'il faut encore décomposer.

Sur le premier schéma (fig. 5-5 a), nous distinguons un ASIC composé de:

- l'interface analogique à laquelle est ajouté un multiplexeur,
- un seul convertisseur A/N,
- une PPU connectée à une RAM,
- un contrôleur.

L'ASIC est relié au capteur de pression, à l'alimentation et à une *EEPROM* (Electrically Erasable Programmable Read-Only Memory) contenant les coefficients utiles à la linéarisation et à la compensation. Les résultats d'une linéarisation et d'une compensation sont transmis à l'extérieur de l'ASIC aux travers de la PPU. Le contrôleur permet de communiquer des informations à l'utilisateur ou de recevoir des ordres.

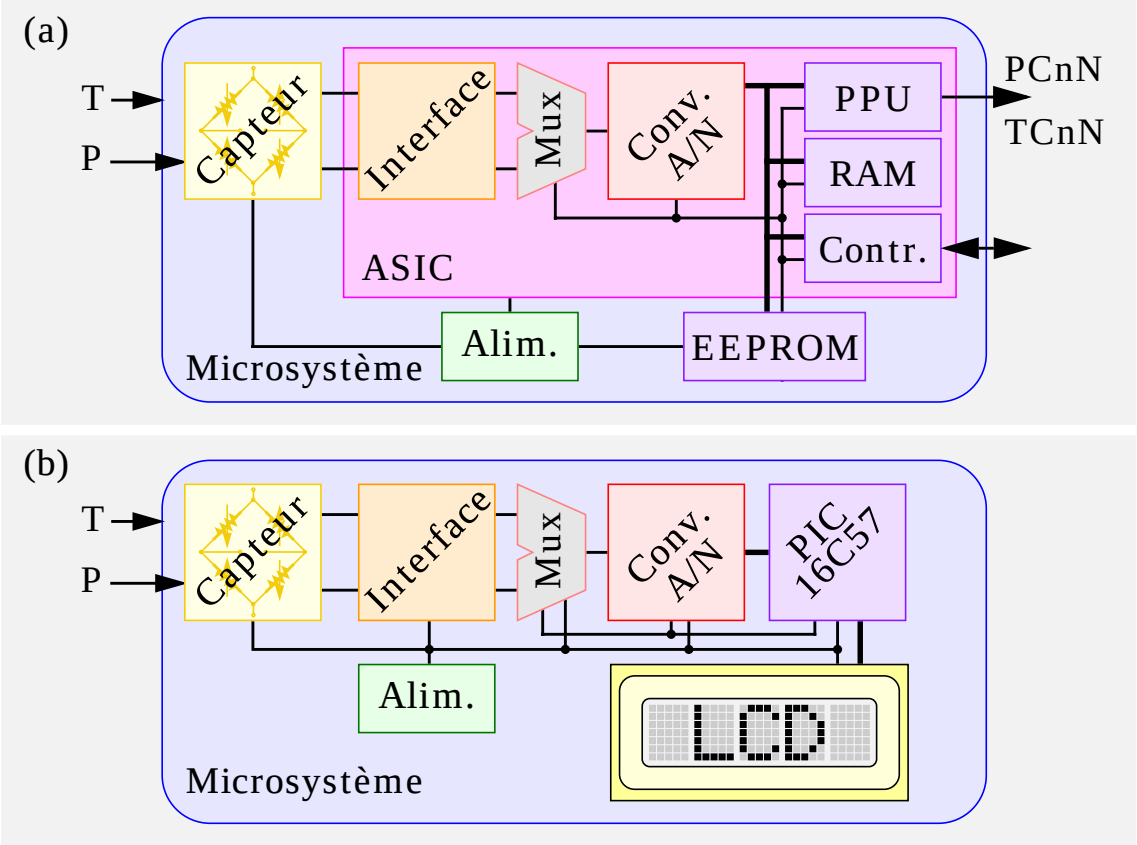


Figure 5-5. (a) Schéma d'une implantation utilisant un ASIC. (b) Schéma d'une implantation utilisant des composants du marché tels qu'un microcontrôleur et un écran LCD.

L'implantation du second schéma (fig. 5-5 b) fait appel à des composants du marché. Sur ce schéma, on distingue:

- l'interface analogique reliée à un multiplexeur,
- un convertisseur A/N,
- un microcontrôleur PIC16C57 de Microchip,
- un écran *LCD* (Liquid Crystal Display),
- l'alimentation.

L'utilisation d'un microcontrôleur sous-entend "répartition de l'algorithme entre le HW et le SW". Comme il s'agit d'un microcontrôleur du marché, cette répartition est imposée. Pour ce microcontrôleur, il n'existe par exemple pas d'instruction assembleur permettant d'effectuer une multiplication. La multiplication doit donc être implantée sous une forme SW. Remarquons que, pour ce

second exemple, l'environnement doit être en mesure d'effectuer de la co-simulation HW-SW.

Lorsque nous avons les schémas blocs des implantations que nous désirons réaliser, nous pouvons commencer les opérations de décomposition successives.

Interface analogique

Il a été décidé d'alimenter le capteur de pression, c'est-à-dire le pont de Wheatstone, avec une tension constante. Il est en effet possible de démontrer [Boe95] que le fait d'alimenter le pont en courant, introduit des non-linéarités d'ordre supérieur plus difficiles à compenser et à linéariser.

Selon notre modélisation du capteur (§ 4.3.2), la résistance équivalente du pont de Wheatstone R(T,0) dépend uniquement de la température. Comme le capteur est alimenté avec une tension constante, nous devons trouver un moyen de mesurer le courant traversant le pont. Celui-ci sera notre mesure de la température. Plusieurs solutions sont envisageables:

- Nous pouvons, par exemple (fig. 5-6 a), placer une résistance R_{T1} en série avec le pont R(T,0). La tension sur cette résistance est la mesure de la température. Pour que la tension sur le pont soit quasi constante, il faut que la valeur de la résistance soit suffisamment petite par rapport à la résistance équivalente du pont. En première approximation, la tension représentant la température de ce schéma est donnée par l'équation (5.2).

$$VT(T) = V_a \cdot \frac{R_{T1}}{R_{T1} + R(T,0)} \cdot \left(1 + \frac{R_{T3}}{R_{T2}}\right) \qquad (5.2)$$

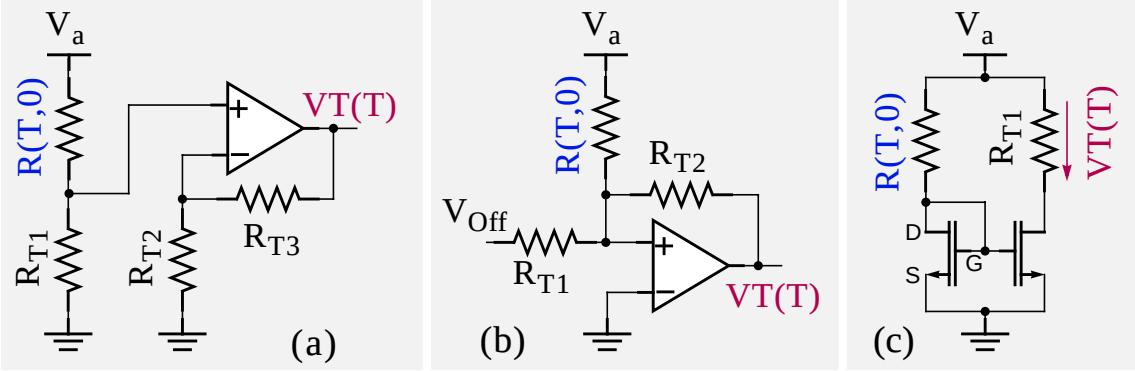


Figure 5-6. Mesure de la température avec la résistance équivalente du pont de Wheatstone. Solutions envisagées.

- Pour éviter d'avoir recours à une résistance en série avec le pont, il est possible de créer une terre virtuelle (fig. 5-6 b). L'avantage de cette solution est que l'on peut effectuer une première correction de la partie DC de la mesure. En première approximation, la tension représentant la température de ce schéma est donnée par l'équation (5.3).

$$VT(T) = -\left(\frac{V_a}{R(T,0)} + \frac{V_{Off}}{R_{T1}}\right) \cdot R_{T2} \qquad (5.3)$$

- Une autre possibilité consiste à utiliser un miroir de courant (fig. 5-6 c). L'inconvénient de cette solution est que les tensions drain-source des transistors dépendent de la température. En première approximation, la tension représentant la température de ce schéma est donnée par l'équation (5.4).

$$VT(T) = \frac{R_{T1}}{R(T,0)} \cdot (V_a - V_{GS}) \quad \& \quad V_{GS} \propto T \qquad (5.4)$$

La sortie du capteur de pression est la sortie différentielle du pont de Wheatstone. Si l'on ne désire pas déséquilibrer électriquement le pont, il faut brancher la sortie à une électronique ayant une impédance d'entrée élevée. Ici également différentes solutions sont envisageables (fig. 5-7).

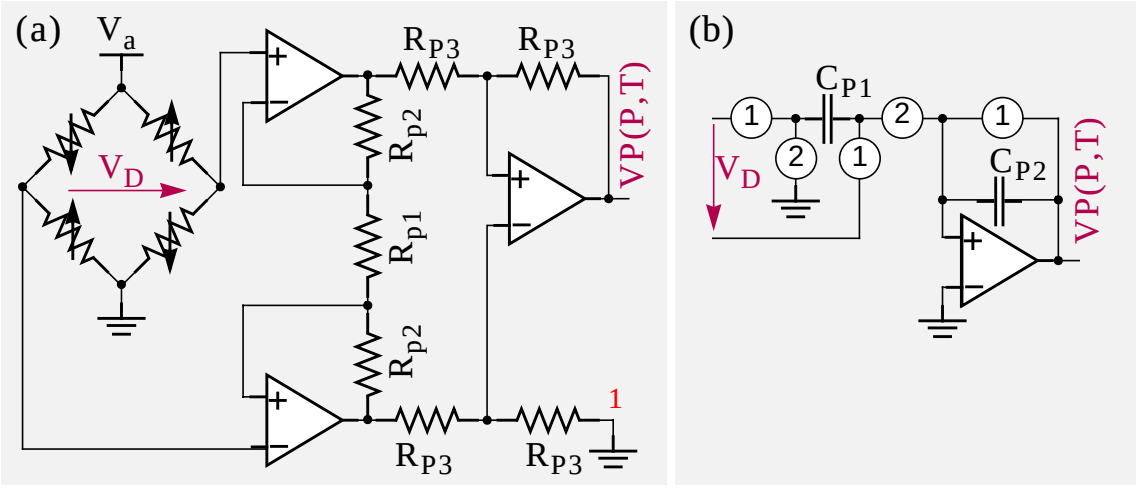


Figure 5-7. Mesure de la pression avec un capteur piézo-résistif, solutions envisagées.

- Le schéma le plus souvent utilisé est l'amplificateur d'instrumentation (fig. 5-7 a). En première approximation, la tension représentant la pression de ce schéma est donnée par l'équation (5.5). Afin d'effectuer une correction de la partie DC de la mesure, il est envisageable de changer le niveau du noeud 1.

$$VP(P,T) = \left(1 + 2 \cdot \frac{R_{P2}}{R_{P1}}\right) \cdot V_D(P,T)) \qquad (5.5)$$

- Une autre possibilité [Boe95] consiste à utiliser des capacités commutées (fig. 5-7 b). En première approximation, la tension représentant la pression de ce schéma est donnée par l'équation (5.6). Pour tenir compte du courant de fuite vers la terre virtuelle, il est possible d'améliorer ce circuit en ajoutant une source de courant [Gee96].

$$VP(P,T) = \frac{C_{P1}}{C_{P2}} \cdot V_D(P,T) \qquad (5.6)$$

Pour l'interface analogique, il existe donc différentes solutions. La simulation permet de tester ces solutions, en utilisant d'abord des éléments idéaux. Ainsi pour la mesure de la température, après des simulations d'éléments idéaux, nous avons choisi d'utiliser le schéma de la figure 5-6 a. Ce n'est qu'ensuite qu'une nouvelle opération de décomposition peut avoir lieu.

Dans le cas de l'ASIC, il s'agit de réaliser les amplificateurs opérationnels avec des transistors (fig. 5-8 b). Dans le cas de la seconde implantation, il s'agit de choisir des amplificateurs opérationnels du marché. Les fabricants de circuits proposent souvent des macro-modèles de leurs composants. La figure 5-8 a présente l'en-tête du macro-modèle de l'amplificateur opérationnel que nous avons utilisé pour la mesure de la température. A nouveau, les simulations permettent d'optimiser et de valider les choix.

Convertisseurs A/N

Les fabricants de circuits intégrés proposent souvent des macro-modèles de leurs composants purement analogiques tels que les amplificateurs opérationnels. En revanche, il est rare de trouver des modèles comportementaux de convertisseurs A/N du marché. Le plus souvent, nous sommes donc contraints de développer nous-mêmes des modèles comportementaux d'éléments d'électronique mixte.

Nous l'avons mentionné au chapitre 3, les simulateurs numériques peuvent être des milliers de fois plus rapides que les simulateurs analogiques. Or les convertisseurs A/N se trouvent aux frontières de ces deux mondes. Il faut donc prendre des précautions afin de ne pas ralentir inutilement la simulation. La précaution principale est qu'il ne faut pas passer inutilement d'un monde à l'autre, c'est-à-dire d'un simulateur à l'autre.

Prenons par exemple le cas de la première implantation (ASIC). Le convertisseur que nous avons choisi d'utiliser est un convertisseur développé à l'IMT [Heu96] [Gri96]. Il s'agit d'un convertisseur A/N dont la dynamique est de 14 bits fonctionnant par approximations successives. Ce composant possède 7 bornes:

- l'alimentation V_{DD},
- l'alimentation V_{SS},
- l'entrée analogique A_{in},
- la terre analogique A_{gnd},

- l'entrée numérique du signal d'horloge Ck,
- l'entrée numérique du signal de synchronisation Sync,
- la sortie numérique de la conversion D_{Out}.

(a)

```
*////////////////////////////////////////////////////////
* (C) National Semiconductor, Inc.
* Models developed and under copyright by:
* National Semiconductor, Inc.

*////////////////////////////////////////////////////////
*LMC6484A CMOS Quad OP-AMP MACRO-MODEL
*////////////////////////////////////////////////////////
*
* connections:       non-inverting input
*                    |   inverting input
*                    |   |   positive power supply
*                    |   |   |   negative power supply
*                    |   |   |   |   output
*                    |   |   |   |   |
*                    |   |   |   |   |
.SUBCKT LMC6484A/NS  1   2   99  50  40
*Features:
*Operates from single or dual supplies
*Rail-to-rail input and output swing
*Ultra low input curre
*Slew rate =
*
*NOTE: Model is for si
*      supply current
*      Noise is not mo
*      Asymmetrical ga
*
*****************INPUT
*
I1   99   4 17U
M1    5   2 4 99 MOSFET
R3    5 50 5.651K
M2    6   7 4 99 MOSFET
R4    6 50 5.651K
...
```

(b)

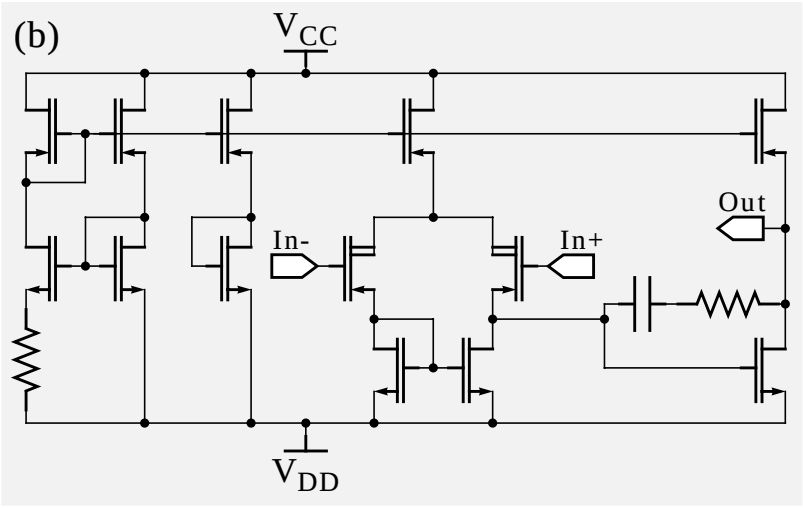


Figure 5-8. Amplificateurs opérationnels utilisés pour la mesure de la température. (a) En-tête d'un macro-modèle du composant LMC6484A de National Semiconducor. (b) Schéma de l'amplificateur utilisant la technologie Alp1lv de EM Micoelectronic-Marin.

La figure 5-9 présente schématiquement trois possibilités de modélisation du convertisseur A/N. Les bornes se trouvant en haut et à gauche du symbole du convertisseur sont des bornes reliées à des stimuli provenant du simulateur analogique. Les bornes se trouvant en bas et à droite sont des bornes reliées au simulateur

numérique. Utiliser une netlist SPICE pour modéliser le convertisseur correspond à la figure 5-9 a. Modéliser une partie du convertisseur avec des éléments SPICE et l'autre partie avec un HDL numérique, correspond à la figure 5-9 b. La dernière figure correspond à une modélisation du convertisseur avec un HDL numérique uniquement. Les temps de simulation les plus courts sont obtenus avec la troisième solution. Inversement, c'est avec la première solution que l'on obtiendra les résultats les plus proches de la réalité. La solution de la figure 5-9 b constitue un bon compromis entre rapidité de simulation et précision [Hom97].

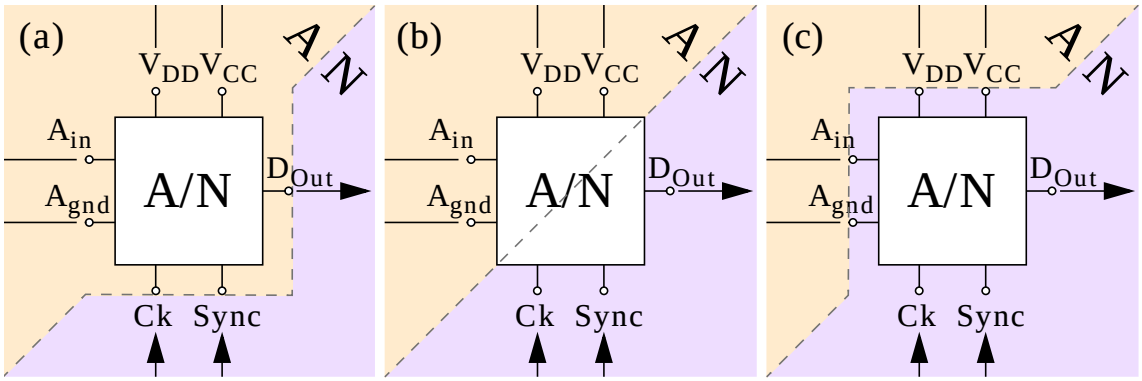


Figure 5-9. Modélisation d'un convertisseur A/N, (a) avec une Netlist SPICE, (b) avec des éléments SPICE et des éléments numériques, (c) avec des éléments numériques uniquement.

Implantation de l'algorithme

Lors de la première phase de l'approche MicroSIM, nous avons déterminé la disposition des PPU à implanter. Nous avons également fixé la représentation des nombres permettant d'atteindre les spécifications. La seconde étape de l'approche MicroSIM est donc consacrée à l'implantation de l'algorithme.

- La figure 5-10 a présente le schéma du microsystème utilisant l'ASIC. On y distingue les symboles du capteur de pression, de l'ASIC, de l'EEPROM, d'une horloge et d'un élément chargé de dénormaliser les résultats. Le but de la dénormalisation est d'obtenir les mêmes unités entre les entrées et les sorties de ce microsystème.

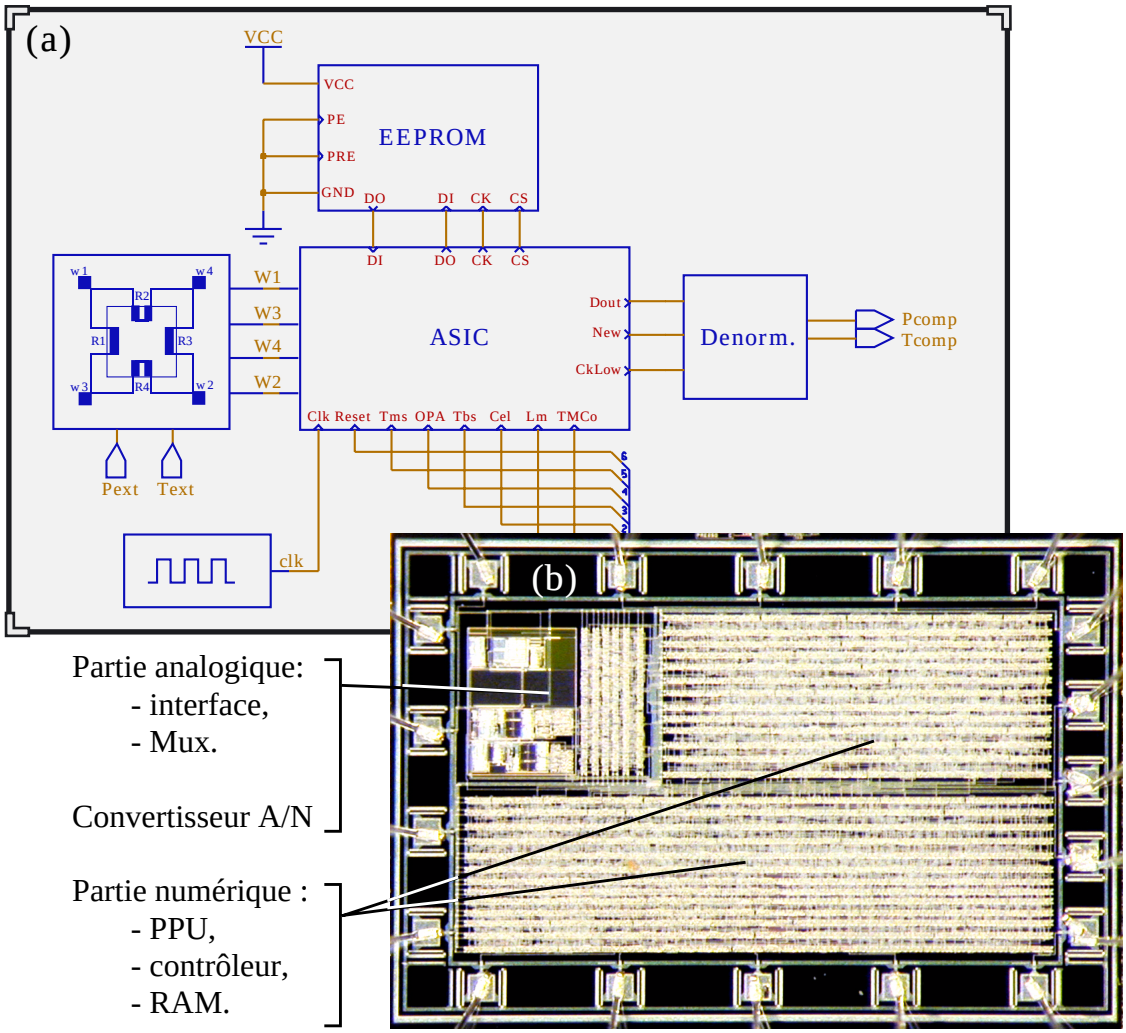


Figure 5-10. (a) Schéma du microsystème utilisant l'ASIC. (b) Photo de d'un ASIC réalisé avec la technologie Alp1lv de EM Microelectronic-Marin.

Lors de l'implantation sur ASIC de l'algorithme, l'approche descendante a été suivie. Du code VHDL a été écrit ou généré puis synthétisé. Les cellules standards ont ensuite été placées puis routées. Après certaines de ces étapes des simulations ont été effectuées. Celles-ci ont permis de contrôler les opérations de décompositions successives. Remarquons que, pour des raisons de librairies, aucune simulation globale n'a pu être entreprise après placement et routage de tous les éléments.

Finalement des prototypes de l'ASIC ont été fabriqués puis testés. La figure 5-10 b est une photo d'un de ces ASIC. Sa taille est de 5.6 mm^2. Il a été réalisé avec une technologie "faible consommation" CMOS de 1µm. Sa consommation est de l'ordre de 150µA pour une alimentation de 3V. La sortie numérique est de 16 bits avec un taux d'échantillonnage de 10kHz.

- Pour l'autre implantation, celle faisant appel à des composants du marché, des simulations globales, incluant la cosimulation HW-SW, ont été effectuées. La figure 5-11 présente le schéma utilisé pour la simulation du microsystème ainsi que la fenêtre permettant de visualiser le software. Sur ce schéma, nous distinguons les symboles:

  - du capteur de pression (HDL-A),
  - de l'interface analogique (éléments SPICE),
  - des convertisseurs A/N (VHDL),
  - du microcontrôleur (VHDL, C, Tcl/Tk),
  - d'un générateur de signal d'horloge (VHDL),
  - d'un écran LCD (VHDL, C, Tcl/Tk).

  Certains de ces modèles utilisent le simulateur analogique alors que d'autres utilisent le simulateur numérique. Il s'agit donc bien d'un exemple de simulation système, multi-natures, multi-modes utilisant la co-simulation HW-SW. Les simulations de ce schéma ont permis d'optimiser et de contrôler le software. Finalement des prototypes [Seg99] de cette implantation ont été fabriqués puis testés et calibrés (fig. 5-11 c & d). Ce microsystème permet de mesurer et d'afficher la température ambiante et la pression absolue.

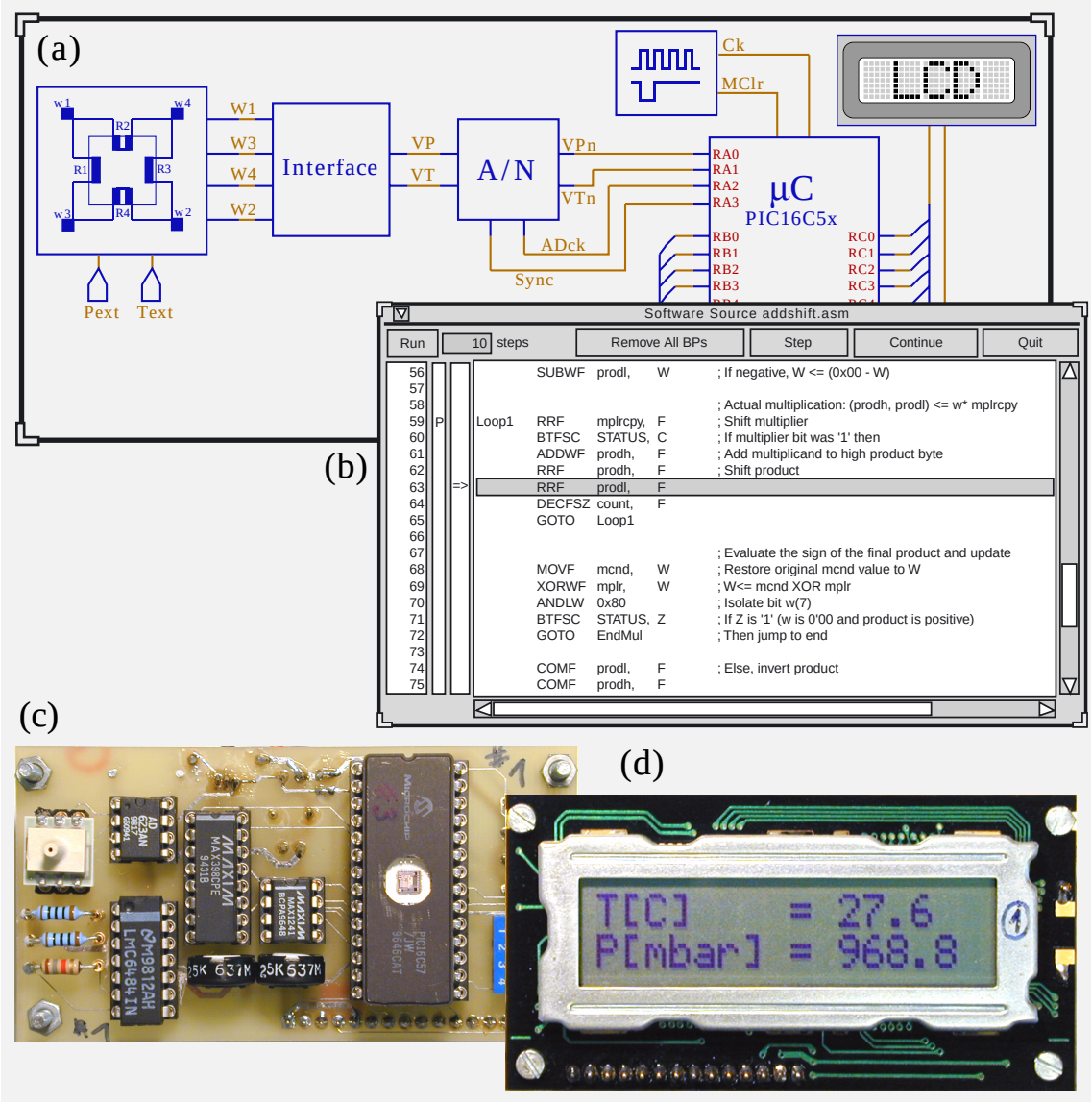


Figure 5-11. Implantation avec des composants du marché. (a) Schéma du microsystème utilisé pour la simulation et (b) fenêtre permettant de visualiser le software. Photos, (c) vue de dessous et (d) affichage LCD.

Discussion

Lors de la première étape de l'approche MicroSIM (modélisation comportementale), nous avons commencé par développer puis valider un modèle comportemental de la famille de capteur que nous devons utiliser. Ensuite nous avons décomposé l'électronique du microsystème en sous-systèmes. Cette première décomposition traite les mesures en parallèle. Cela a pour principal avantage

d'offrir la possibilité de ne pas introduire de délais entre les entrées et les sorties du microsystème. Ne pas introduire de délais entre les entrées et les sorties n'est certes pas réaliste. C'est une manière de concentrer les efforts de conception sur des problèmes algorithmiques tels que: l'ordre des PPU, la représentation finie des nombres, etc. Durant cette première étape, c'est un simulateur analogique de type SPICE qui a été utilisé. Les simulations effectuées nous ont permis de répondre aux principales questions que nous nous posions, et par là même, ont permis d'optimiser l'architecture. Lors de cette première phase, nous nous sommes également intéressés au calibrage du microsystème, ce qui nous a permis d'en optimiser les procédures.

La seconde étape (modélisation post-design) a été consacrée à deux implantations différentes. L'une a débouché sur la réalisation d'un ASIC alors que l'autre fait appel à des composants du marché. Pour chacune de ces implantations, un seul convertisseur A/N est utilisé. Le traitement des mesures est donc effectué de façon séquentielle suivant des FSM. Lors de l'implantation sur ASIC, l'approche MicroSIM nous a permis de garder le contrôle du développement. Nous avons ainsi pu corriger des erreurs que nous n'aurions peut-être pas détectées avant la réalisation de prototypes. Par exemple l'implantation de la multiplication dans les PPU comportait une erreur de troncation. Voici en quelques mots l'erreur qui avait été commise et ce qui a permis de la corriger.

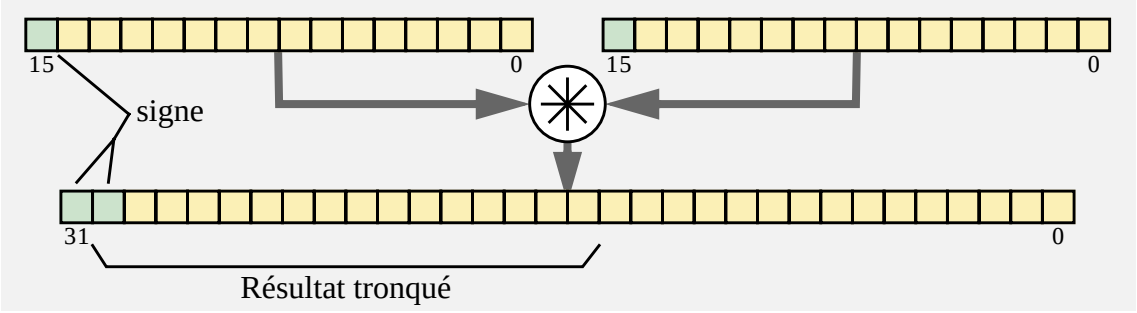


Figure 5-12. Multiplication de deux nombres binaires signés de 16 bits avec troncation.

Nous avons vu qu'une PPU effectue une succession d'additions et de multiplications. Nous avons donc dû implanter la multiplication de deux nombres binaires signés de 16 bits dont le résultat est

tronqué de manière à ne prendre que 16 bits. En réalité le résultat exact d'une telle multiplication utilise 32 bits. L'erreur que nous avons commise se situe au niveau de la troncation de ces 32 bits. En effet, nous avons tronqué les 16 bits de poids faible. Or, avec la représentation que nous avons choisie (complément à deux), les deux bits de poids forts du résultat d'une multiplication sont toujours les mêmes, il s'agit d'un doublon du signe (fig. 5-12). On imagine aisément les conséquences de cette erreur sur le résultat d'une succession de multiplications! Ce sont des simulations globales du microsystème qui ont révélé cette erreur.

Lors de la seconde implantation, celle faisant appel à des composants du marché, la co-simulation HW-SW a pu être expérimentée. Elle nous a permis de tester le SW et le HW en effectuant de véritables analyses transitoires. Ceci, bien évidemment avant que des prototypes existent!

5.2 Second exemple, mesure du pH et de la conductivité

Le second exemple que nous avons choisi de présenter est celui de capteurs miniatures en silicium reliés à une électronique, à un écran LCD et à une source d'énergie telle que des batteries boutons. Le tout forme un instrument de mesure de la taille d'un gros stylo. Comme pour l'exemple présenté au paragraphe précédent, ce dispositif correspond à notre définition d'un microsystème. Les capteurs de ce microsystème sont au nombre de trois:

1. une diode pour la mesure de la température,
2. un *ISFET* (Ion-Sensitive Field Effect Transistor) pour la mesure du pH,
3. une cellule à quatre électrodes pour la mesure de la conductivité.

A l'exception de l'électrode de référence de l'ISFET, ces trois capteurs prennent place sur une seule pièce de silicium (fig. 5-13).

La réalisation d'un tel instrument de mesure est étroitement liée aux caractéristiques des capteurs. Il faut donc que celles-ci

aient été étudiées et comprises. Pour ces trois capteurs, les non-linéarités, les dérives temporelles, le vieillissement, la dépendance à la température, l'interdépendance des capteurs plongés dans le même liquide, sont autant de caractéristiques dont il faut tenir compte lors du design de l'électronique. En plus, il faut tenir compte des variations des caractéristiques des capteurs lorsqu'ils sont produits en "grandes" quantités. Par exemple la large dispersion du point de fonctionnement des ISFET. Remarquons qu'à ces contraintes technologiques viennent généralement s'ajouter des contraintes économiques de production. Or celles-ci sont parfois incompatibles avec les spécifications.

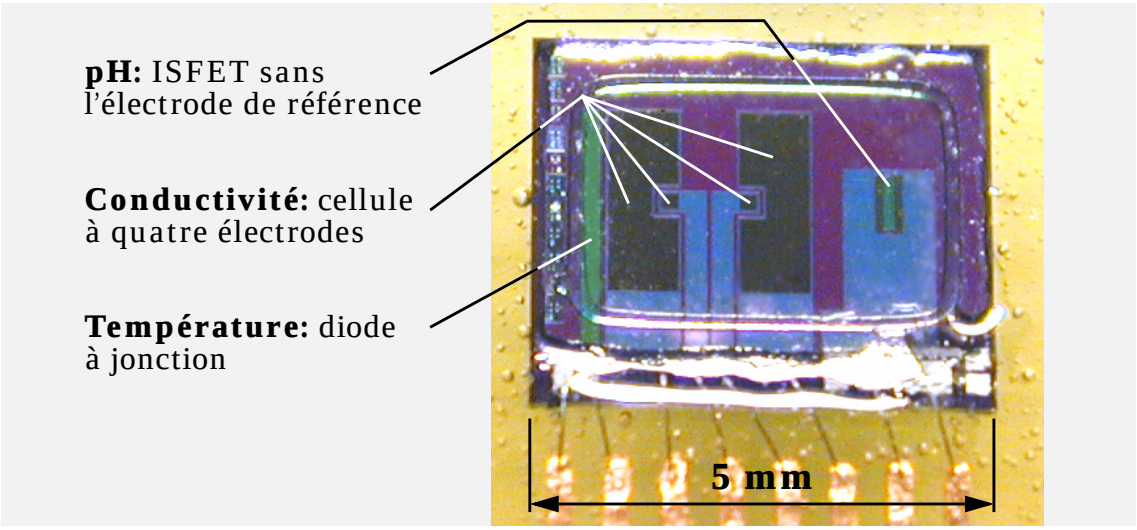


Figure 5-13. Photo des capteurs miniaturisés.

Nous l'avons mentionné précédemment, avant de commencer la première phase de l'approche MicroSIM, il faut disposer de spécifications. Pour ce second exemple, il est par exemple exigé de concevoir un microsystème formé de plusieurs parties: la pointe, le corps et la source d'énergie (les piles boutons). En effet, l'électrode de référence de l'ISFET a une durée de vie limitée. Après un certain temps, il faut que l'utilisateur du microsystème puisse la changer. Une autre caractéristique de ce type de capteur de pH, est sa lente dérive. Après un certain nombre de mesures, l'utilisateur doit avoir la possibilité de calibrer l'ISFET. Pour d'autres raisons, l'utilisateur doit également avoir la possibilité de calibrer la cellule à quatre électrodes et l'électronique.

Dans les paragraphes suivants, nous allons nous concentrer sur la première phase de l'approche MicroSIM, à savoir la modélisation comportementale. En effet, nous n'avons pas suivi l'approche MicroSIM dans sa seconde phase (modélisation post-design). Nous verrons plus loin pourquoi (c.f. § 5.2.3).

5.2.1 Modélisation comportementale

Conformément aux buts de la première phase de l'approche MicroSIM, nous avons cherché à décomposer le microsystème en sous-systèmes. La figure 5-14 présente le résultat de cette opération de décomposition. Nous y retrouvons la décomposition en trois parties (pointe, corps et source d'énergie) imposée par les spécifications.

La pointe est composée des trois capteurs et d'une mémoire (EEPROM) servant à recevoir des informations relatives aux capteurs. Le corps est composé d'alimentations, d'une interface analogique, d'un convertisseur A/N, d'un microcontrôleur (µC), d'une mémoire (EEPROM), d'un bouton-poussoir et d'un écran LCD.

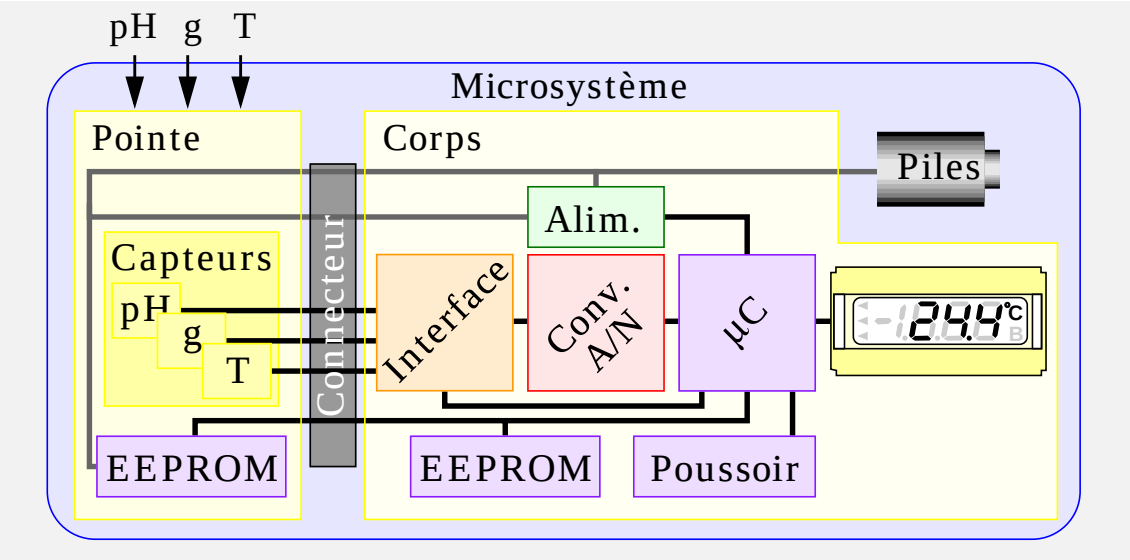


Figure 5-14. Décomposition du microsystème en sous-systèmes.

Les stimuli du microsystème sont le pH, la conductivité (g) et la température (T). Tous trois sont appliqués à la pointe du microsystème. En réalité, la température est également transmise aux

corps du microsystème, ce qui donne lieu à un gradient de température entre la pointe et le corps. Nous n'en avons pas tenu compte lors de cette première phase de conception.

L'interface utilisateur est constituée du bouton-poussoir et de l'écran LCD. C'est-à-dire que toutes les opérations de mesures et de calibrage sont entreprises par l'intermédiaire d'un unique bouton-poussoir.

Nous l'avons mentionné, les caractéristiques des capteurs varient lorsqu'ils sont produits en grande quantité. Or, les caractéristiques de l'électronique du corps varient également lors de la production. C'est pour cette raison que des mémoires ont été ajoutées dans les pointes et dans les corps. Ainsi lorsqu'un corps est produit, il est caractérisé. De cette caractérisation sont extraits des coefficients qui sont introduits dans la mémoire du corps. On procède de façon analogue avec les pointes. Finalement, lorsqu'une pointe et un corps sont connectés, le microcontrôleur utilise les coefficients se trouvant dans les deux mémoires pour traiter les mesures.

L'utilisateur peut à tout moment effectuer un calibrage en plongeant les capteurs dans une solution dont le pH ou la conductivité sont connus. Après cette opération de calibrage, les coefficients se trouvant dans la mémoire de la pointe sont actualisés.

La figure 5-14 est la décomposition finale du microsystème en sous-systèmes. La figure 5-15 est un diagramme bloc du microsystème sans alimentation, sans bouton-poussoir et sans affichage LCD. Ce diagramme indique la façon dont sont implantés les algorithmes permettant de traiter les mesures. Comme pour l'exemple précédent, cette première décomposition traite les mesures en parallèle. Nous pouvons constater que certains coefficients utiles aux algorithmes proviennent de la mémoire de la pointe alors que d'autres proviennent de la mémoire du corps. Les sorties de ce diagramme sont: la température linéarisée (Tc), le pH compensé et linéarisé (pHc) et la conductivité compensée (gc).

Le fonctionnement des trois capteurs est le suivant. Pour mesurer la température, un courant est injecté dans la diode et ainsi la tension aux bornes de la diode est une mesure de la température (cf. § 4.3.1).

Pour mesurer le pH, le potentiel de l'électrode de référence est fixé ainsi que le courant et la tension Drain-Source de l'ISFET [Arn88] [Vle88]. La tension de la Source est une fonction du pH et de la température.

La mesure de la conductivité est une mesure à quatre points. Le capteur est donc formé de quatre électrodes, deux grandes et deux petites. Pour mesurer la conductivité, une tension alternative est appliquée aux deux grandes électrodes et le courant injecté est mesuré. La tension apparaissant sur les deux petites électrodes est également mesurée. Ces deux mesures autorisent le calcul de la conductivité du liquide en contact avec les électrodes.

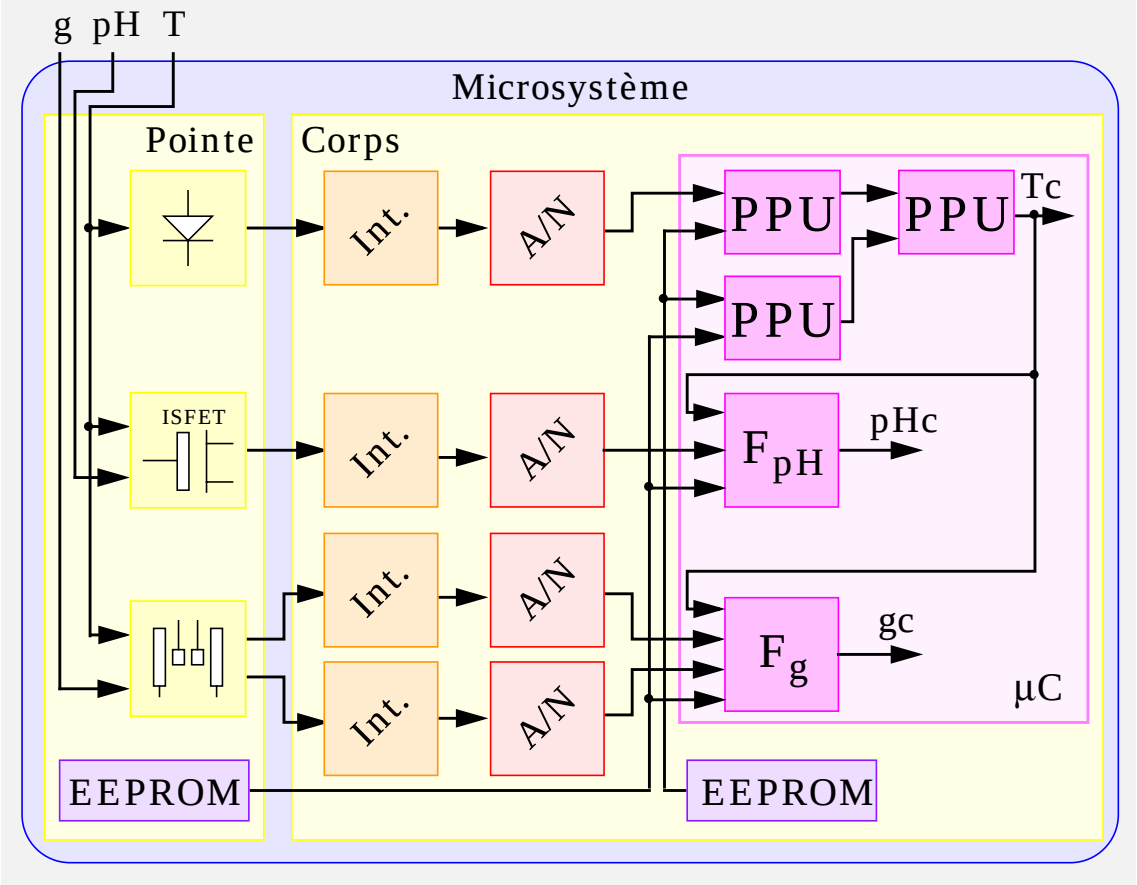


Figure 5-15. Diagramme bloc du microsystème (sans l'alimentation, le bouton-poussoir et l'écran LCD).

L'interface analogique est chargée d'amener les différents capteurs à leur point de travail et d'effectuer une première mise en forme des mesures. Les signaux sont ensuite numérisés par un convertisseur A/N.

Conformément à la première phase de l'approche MicroSIM, nous avons développé des modèles comportementaux pour chacun des blocs de la figure 5-15. Nous avons ainsi pu développer et tester les algorithmes numériques nécessaires au traitement des mesures. Prenons l'exemple de la température. Nous l'avons mentionné, pour mesurer la température, nous utilisons une diode. Nous avons également vu que les équations de la diode idéale utilisée dans le sens direct définissent une surface (fig. 4-4 a). Cette surface sera légèrement différente pour chacune des diodes produites.

Pour obtenir une mesure de la température, nous injectons un courant dans la diode et nous mesurons la tension. Or chaque source de courant, chaque interface analogique et chaque convertisseur A/N des corps produits ont des caractéristiques légèrement différentes. Comment tenir compte de ces différences de manière à ce qu'une pointe et un corps quelconque puissent donner des mesures correctes de la température?

Comme le montre la figure 5-15, afin de résoudre ce problème, nous avons également adopté l'approche polynômiale. Sur cette figure sont représentés trois PPU. La première (en haut à gauche) est alimentée avec des coefficients caractérisant l'interface analogique et le convertisseur A/N. Cette première unité permet de corriger les mesures de manière à tenir compte des imperfections de l'interface analogique et du convertisseur A/N. Il reste à tenir compte des caractéristiques de la diode et de la source de courant. C'est exactement ce dont sont chargées les deux unités restantes.

Pour traiter les mesures du pH et de la conductivité, nous avons utilisé des algorithmes plus traditionnels imposés par notre partenaire industriel. Dans tous les cas, la simulation globale nous a permis d'étudier le bruit de quantification ainsi que l'influence de la représentation finie des nombres dans le domaine numéri-

que. Nous avons ainsi pu déterminer les ressources nécessaires au niveau de l'interface analogique, de la conversion A/N et des algorithmes de traitement des mesures. Nous avons également pu étudier et définir les procédures de calibrage des corps et des pointes. Que ce soit lors de la production des éléments ou par l'utilisateur du microsystème. A la fin de cette première étape de conception, nous disposons donc de tous les éléments pour réaliser l'implantation HW et SW de ce microsystème.

5.2.2 Implantation HW et SW

Lors de la phase de modélisation comportementale, nous avons pu déterminer les ressources nécessaires pour atteindre les spécifications. La seconde phase est donc consacrée à l'implantation. Mais, avant d'aller plus loin, nous devons placer ce travail dans son contexte. Le développement et la réalisation de ce microsystème nous ont été demandés par un des leader de la fabrication d'instruments de mesures de produits chimiques. Actuellement, celui-ci propose des instruments de mesures du pH, de la conductivité et de la température dont les volumes sont relativement importants (de l'ordre de 2000 cm^3) et dont les capteurs ne sont pas miniaturisés.

La figure 5-13 montre qu'au niveau des capteurs, des efforts de miniaturisation ont été entrepris. Notre travail est donc de miniaturiser, voire de simplifier l'électronique. Traditionnellement, pour mesurer la conductivité, une tension alternative de forme sinusoïdale est utilisée. Dans le but de simplifier l'électronique, nous avons décidé de ne pas utiliser une tension de forme sinusoïdale mais de forme rectangulaire. A notre connaissance, cette façon de procéder n'a jamais été tentée. Or ce changement a des implications non triviales du côté chimique. C'est pour cette raison que nous ne pouvons pas nous contenter de simulations pour valider ce changement. Des tests en laboratoire sont absolument nécessaires! Nous avons donc décidé de ne pas suivre l'approche MicroSIM dans sa seconde phase et de réaliser un prototype sans tenir compte des dimensions finales. La figure 5-16 est une photo du prototype réalisé. On y distingue les différents sous-systèmes de la décomposition du microsystème.

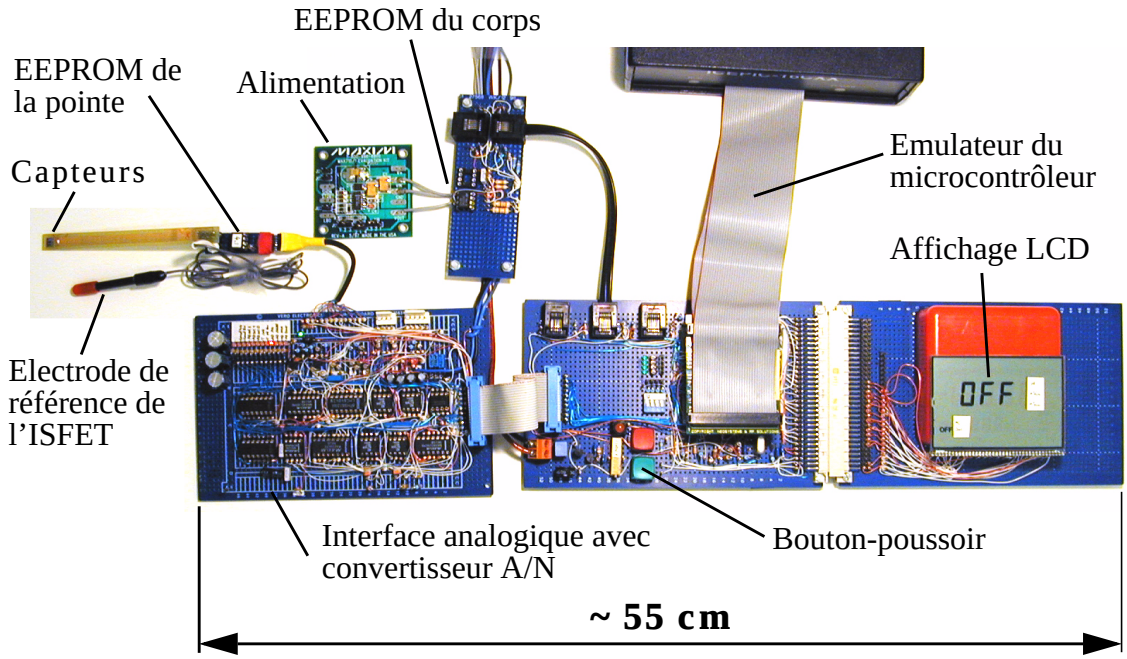


Figure 5-16. Photo du prototype du microsystème.

Pour développer et déboguer le SW, nous avons utilisé un compilateur C ainsi qu'un émulateur de microcontrôleur connecté à un PC. C'est au laboratoire de chimie que nous avons testé le système. Les tests effectués nous ont permis de mieux comprendre le fonctionnement des capteurs et d'améliorer l'électronique. Ce sont ces tests qui nous ont permis de considérer le bruit du système. Les sources de bruit les plus importantes se situent aux niveaux des capteurs, de l'alimentation (amplification de la tension des piles) et de l'électronique numérique. Afin d'augmenter le SNR, nous avons ajouté des filtres numériques dans la chaîne d'acquisition et de traitement des mesures. Finalement, les tests au laboratoire (de nombreuses heures) nous ont permis de valider l'électronique, c'est-à-dire le HW, ainsi que le SW.

Le schéma électrique étant validé, la prochaine étape est sa miniaturisation, opération que nous avons sous-traitée. La figure 5-17 présente le résultat de cette opération. Nous y distinguons la pointe formée des capteurs (sans l'électrode de référence) et de l'EEPROM, puis le corps avec tous les composants électroniques ainsi que l'écran LCD, et le bouton-poussoir. Le *PCB* (Printed Circuit Board) du corps est un PCB de quatre couches. Nous pou-

vons voir que la densité des composants y est importante. Certains éléments n'ont d'ailleurs pas de boîtiers. Ils sont collés à même le PCB avant d'être reliés à celui-ci puis recouverts par une couche protectrice.

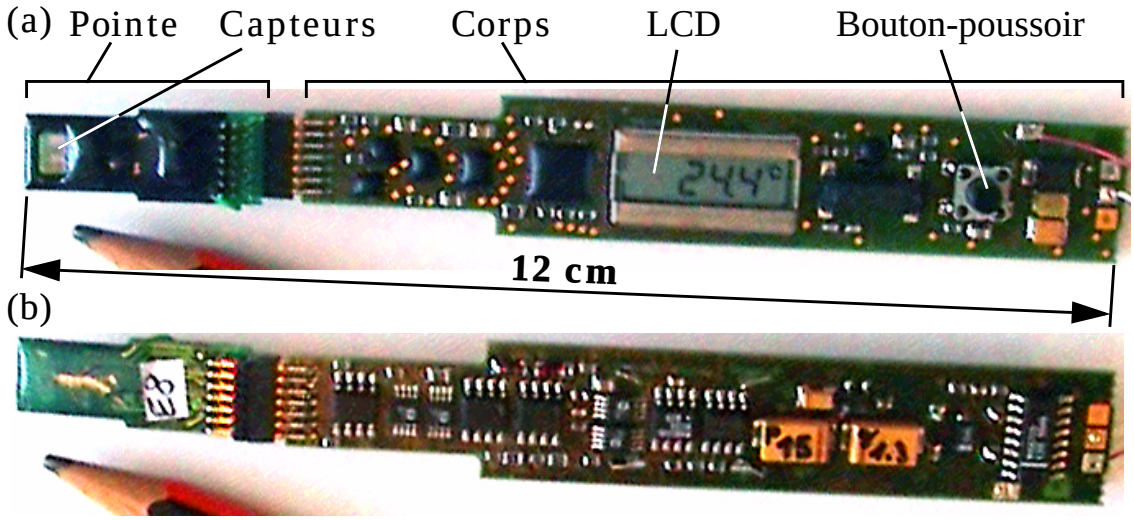


Figure 5-17. Photos du microsystème sans l'électrode de référence, (a) vue de dessus, (b) vue de dessous.

Après sa fabrication, il faut encore programmer le microcontrôleur, programmer l'EEPROM du corps et calibrer l'électronique. On peut ensuite calibrer une pointe et la connecter au corps. Le microsystème est alors prêt à fonctionner. Les tests en laboratoire ont montré que le microsystème fonctionne correctement, c'est-à-dire comme le prototype.

5.2.3 Discussion

Pour ce second exemple, la première phase de l'approche Micro-SIM a été suivie. Le microsystème a d'abord été décomposé en sous-systèmes. Ensuite des modèles comportementaux des sous-systèmes ont été développés. Le développement des modèles a commencé par les capteurs et a continué en suivant le chemin pris par les mesures. Comme pour le premier exemple de réalisation, et pour les mêmes raisons, cette première décomposition traite les mesures de façon parallèle. C'est un simulateur de type SPICE qui a été utilisé. Les simulations ont permis une première optimisa-

tion du HW, des algorithmes de traitement des mesures et des procédures de calibrage.

Afin de miniaturiser l'électronique, nous avons décidé de la simplifier. Parce que nous manquons d'expérience et de connaissances dans la mesure de conductivité, les simplifications envisagées ont dû être validées en laboratoire. C'est pour cette raison que la seconde phase de l'approche MicroSIM n'a pas été suivie.

Au lieu de rester dans l'environnement de simulation afin d'implanter le HW et le SW de ce microsystème, il a été décidé de réaliser un prototype ne tenant pas compte de l'encombrement final. Celui-ci a permis de valider les simplifications envisagées, et plus généralement le choix des composants ainsi que le schéma électrique. Remarquons que la simulation globale nous aurait également permis de valider le schéma, sans pour autant valider les simplifications.

Au laboratoire ce sont de nombreuses heures qui ont été nécessaires aux expériences de validations. En effet, hormis le fait que les liquides utilisés doivent avoir un pH ou une conductivité déterminée, il faut que ceux-ci se trouvent à une température stabilisée et contrôlée. Tout changement de température demande donc du temps. Cette approche nous a permis de mieux comprendre le fonctionnement des capteurs et d'étudier le bruit des mesures, ce qui a débouché sur des améliorations du HW et du SW.

Finalement, des prototypes miniaturisés ont été produits puis testés en laboratoire. Nous avons ainsi pu constater que le fonctionnement de ceux-ci est conforme à ce qui était attendu. Le but de ce travail était de démontrer la faisabilité d'un tel produit avec des contraintes de précision, d'encombrement et de consommation relativement importantes. Ce but a été atteint, en partie, grâce à l'approche MicroSIM.

5.3 Conclusion

Dans ce chapitre, nous avons présenté deux exemples de réalisations de microsystèmes. Ceux-ci sont composés de capteurs et d'électronique (analogique et numérique), mais pas d'actionneurs. Les capteurs sont, dans le premier cas, un capteur de pression et de température, et dans le second cas, des capteurs de pH, de conductivité et de température. Dans les deux cas, l'électronique est chargée d'alimenter les capteurs et de traiter les mesures avec une bande passante relativement peu élevée. Il s'agit donc d'exemples particuliers.

Il est cependant intéressant de noter que, pour le premier exemple, les deux phases de l'approche MicroSIM ont été suivies, alors que pour le second, ce n'est que la première phase qui a été suivie. La seconde phase n'a pas été suivie à cause d'un manque connaissances concernant la mesure de la conductivité. Ce qu'il est important de retenir, c'est que pour suivre l'approche MicroSIM, il faut disposer de modèles de microstructures validés et suffisamment réalistes. Dans le cas du second exemple, il aurait été possible de suivre la seconde phase si nous avions disposé d'un modèle plus réaliste du capteur de conductivité. Au lieu de cela, de nombreuses heures de laboratoire ont été nécessaires à l'acquisition des connaissances manquantes.

Il serait plus simple que les personnes développant les microstructures prennent l'habitude de créer des modèles comportementaux afin de les transmettre à ceux qui sont chargés de développer l'électronique. C'est, à notre sens, une des clés pour que l'approche MicroSIM soit applicable.

Nous pouvons également noter que lors de la première phase de l'approche MicroSIM, et pour ces deux exemples de réalisations, le traitement des mesures est effectué de façon parallèle, quand bien même le résultat de l'implantation est un traitement séquentiel des mesures. Dans le cas particulier de ces exemples, c'est une manière de concentrer les efforts de conception sur certains problèmes algorithmiques. Durant cette première étape c'est un simulateur analogique de type SPICE qui a été utilisé pour réaliser des

simulations transitoires. Finalement la première étape de l'approche MicroSIM a permis une optimisation de l'architecture en vue d'atteindre les spécifications. Elle a également permis d'appréhender le problème du calibrage.

5.4 Références

[Arn88] Ch.Arnoux, *"Fabrication et étude des phénomènes de dérive de capteurs du type pH-ISFET a membrane en oxyde d'aluminium"*, thèse de doctorat, Université de Neuchâtel, Suisse, 1988

[Boe95] A.Boegli, *"Compensation de capteurs de pression piézo-résistifs"*, Rapport technique No 705, CSEM, 1995

[Gee96] B.De Geeter, O.Nys, J.-P.Bardyn, *"A wide temperature range micropower sensor interface circuit"*, ESSCIRC'96, Proceeding of 22nd European Solid-State Circuits Conference, Neuchâtel, 1996, pp.136-139

[Gri96] L.Grisoni, A.Heubi, P.Balsiger, F.Pellandini, *"Micro Power 14 bits RSD A/D Converter"*, ICSPAT'96, Boston, 1996, p.510-514

[Heu96] A.Heubi, P.Balsiger, F.Pellandini, *"Micro Power 13 bits Cyclic RSD A/D Converter"*, ISLPED'96, Monterey, USA, 1996, pp.253-257

[Hom97] W.Homan, *"An FPGA-based hardware demonstrator for the digital signal processing of audio signals"*, Rapport interne, IMT 420 PE 07/97, Institut de Microtechnique, Université de Neuchâtel, Suisse

[Seg99] U.Seger, *"Réalisation pratique d'un système de petites dimensions"*, Projet de semestre d'hiver 1998/99, Institut de Microtechnique, Université de Neuchâtel, Suisse

[Vle88] H.H.van den Vlekkert, *"Some fundamental and practical aspects of CHEMFETs"*, Thèse de doctorat, Université de Neuchâtel, Suisse, 1988

Chapitre 6

Conclusion

Dans ce travail, nous avons présenté les deux principaux types de conception à savoir descendant et ascendant. Nous avons pu nous rendre compte que la conception de type descendante offre de nombreux avantages. Certaines parties d'un microsystème, à savoir l'électronique numérique, l'électronique analogique et le software, peuvent être conçues en suivant l'approche descendante. Ce n'est pas le cas des microstructures qui sont conçues en suivant le type de conception ascendant.

La méthode de conception globale de microsystèmes que nous présentons dans ce travail tient compte de cette constatation. Il s'agit de l'approche MicroSIM. Celle-ci se base sur un environnement de CAO particulier.

Pour ce travail, nous avons utilisé différents outils de CAO existants que nous avons adaptés. Notre design-kit pour microsystème contient ces adaptations. Les exemples de réalisations présentés au chapitre 5 démontrent que la méthode permet effectivement de réaliser des microsystèmes avec un grand contrôle.

6.1 Contributions

6.1.1 Approche MicroSIM

Le but de l'approche MicroSIM est d'offrir une méthode pour réaliser des microsystèmes, et de promouvoir un environnement CAO pour microsystèmes. La méthode exposée dans ce travail est de type descendante. Elle peut être décomposée en deux étapes: la modélisation comportementale et la modélisation post-design. Les principaux avantages de cette méthode sont:

1. d'offrir un grand contrôle du développement,
2. de démontrer la faisabilité d'un microsystème très tôt dans le processus de développement.

Mais, pour être en mesure d'appliquer cette méthode, il faut disposer d'un environnement de CAO particulier.

6.1.2 Outils de CAO

Au niveau de l'environnement de CAO, les contraintes sont importantes. Premièrement des design-kits possédants de "bons modèles" doivent être disponibles. Deuxièmement, les simulateurs doivent admettre des simulations globales, multi-natures, multi-modes, multi-niveaux ainsi que la co-simulation HW-SW. A notre connaissance, il n'existe pas de simulateur conçu pour admettre tous ces types de simulation. Avec l'arrivée de la norme VHDL-AMS, de nouveaux simulateurs font leur apparition. Il est probable que certains de ceux-ci répondent aux exigences mentionnées dans ce travail.

En nous intéressant à la physique et aux mathématiques permettant de modéliser les sous-systèmes, nous avons constaté que:

1. L'électronique analogique peut être modélisée par un système à constantes localisées et simulée avec un simulateur de type SPICE.

2. L'électronique numérique peut être modélisée de façon plus approximative en utilisant un HDL tel que VHDL et simulée avec un simulateur commandé par des évènements.
3. Les microstructures sont généralement des systèmes à constantes réparties. Elles peuvent être approchées par des systèmes à constantes localisées, modélisées en utilisant un langage tel que HDL-A ou VHDL-AMS et simulées avec un simulateur de type SPICE.
4. Les processeurs utilisés dans les microsystèmes sont généralement de petits processeurs RISC. Ils peuvent être modélisés par un ISS couplé à une ROM et simulé avec un simulateur commandé par des évènements.

6.1.3 Elaboration d'un design-kit

Pour ce travail, de manière à être en mesure de suivre l'approche MicroSIM, nous avons choisi d'utiliser un environnement développé pour la CAO d'électronique et nous avons développé un design-kit. Celui-ci contient une bibliothèque de modèles de microstructures ainsi qu'un ISS d'un microcontrôleur et un générateur de ROM. Lors de l'élaboration de certains de ces modèles, des difficultés ont été éprouvées. Nous retenons de cette expérience que pour construire de "bons modèles", il ne suffit pas de connaître le langage de modélisation, il faut également connaître le simulateur.

6.1.4 Réalisation de microsystèmes

Les exemples de réalisations présentés dans ce travail montrent que cette méthode:

1. incite les spécialistes des différents domaines à communiquer entre eux,
2. permet de démontrer la faisabilité d'un microsystème très tôt dans le processus de développement.

3. permet de garder un grand contrôle du développement d'un microsystème.

Nous avons également vu que pour étudier un algorithme avant son implantation, c'est-à-dire lors de la première étape de l'approche MicroSIM, il est possible d'utiliser un simulateur analogique de type SPICE. Un des avantages de ce choix est qu'il autorise l'exécution de simulation de type AC.

6.1.5 Transfert du savoir-faire

Dans les chapitres 4 & 5, nous avons vu que certains modèles de microstructures ont été développés puis utilisés dans des projets. L'approche MicroSIM a également été mise en pratique dans le cadre de projets de recherches, de projets d'étudiants et de projets avec des entreprises industrielles. Un savoir-faire a donc été développé puis transmis à des partenaires industriels ainsi qu'à des membres de l'IMT et à des étudiants.

6.2 Limitations

La méthode MicroSIM comporte également des limitations. L'une d'elles est le temps de simulation. Lorsque les modèles des sous-systèmes deviennent détaillés, les temps de simulation peuvent devenir extrêmement longs. Il faut alors trouver le bon compromis entre le niveau de détail et le temps de simulation. Parfois également, un microsystème peut posséder des constantes de temps très différentes. C'est le cas lorsqu'une partie d'un sous-système travaille vite alors qu'une autre partie travaille lentement. Dans un tel cas les temps de simulations explosent et cette approche devient plus difficile à appliquer.

6.3 L'avenir

Dans le futur, il faut continuer à développer des bibliothèques de modèles de microstructures. L'idéal serait que les spécialistes des microstructures apprennent à créer de "bons modèles" de leurs

microstructures. Cela faciliterait grandement le travail de ceux qui sont chargés de développer l'électronique. Que ce soit dans le cadre de la conception de microsystèmes ou de systèmes plus grands.

Remerciements

Ce travail a été réalisé à l'Institut de Microtechnique dans le laboratoire d'électronique et de traitement du signal (ESPLAB) sous la direction du Professeur M. F. Pellandini, que je tiens à remercier pour m'avoir donné le soutien et l'indépendance nécessaires à l'aboutissement de cette thèse. Je remercie également M. le Professeur N. de Rooij qui m'a permis d'utiliser les infrastructures du laboratoire des capteurs, actionneurs et microsystèmes (SAMLAB).

J'exprime ma gratitude au Dr H.P. Amann, qui m'a soutenu et encouragé à réaliser ce travail. Sa confiance et sa patience m'ont aidé à débuter et mener à bien ce travail, qu'il a suivi dans son ensemble.

Je tiens également à remercier le Dr V. Moser, qui m'a fait découvrir la modélisation et la simulation. Sa rigueur scientifique m'a aidé à m'exprimer à travers cette thèse, qu'il a également suivie. Nos longues discussions - qui ont souvent agréablement dérivé vers d'autres sujets passionnants - ont permis de clarifier les parties essentielles de l'approche présentée et d'améliorer de nombreux aspects et détails rédactionnels.

Je remercie aussi tous ceux qui, de près ou de loin, ont contribué à la réalisation de ce travail. Je pense ici particulièrement au Dr P. van der Wal, au Dr R. Riem-Vis, au Dr G.-A. Racine, à Mme

S. Pata, à M. Ch. Berthoud et à M. D. Daudet avec lesquels j'ai eu du plaisir à collaborer.

Je remercie les membres du Jury pour le temps qu'ils ont consacré à la lecture du texte et à la critique de mon travail.

Enfin, j'ai une pensée pour ma famille qui m'a soutenu durant les années qu'aura duré ce travail.

Annexes A

Développements mathématiques

A.1 Approximation du quotient différentiel

Le quotient différentiel peut être approché de la manière suivante:

$$sX(s) \;\bullet\!\!-\!\!\circ\; \frac{d}{dt}x(t) \cong \frac{x(t+T)-x(t)}{T} \qquad (A.1)$$

En échantillonnant le signal avec une période de T,

$$x(kT) = x_k \qquad (A.2)$$

nous obtenons:

$$\frac{x_k - x_{k-1}}{T} \;\square\!\!-\!\!\blacksquare\; X(z)\cdot\frac{1}{T}\cdot(1-z^{-1}) \qquad (A.3)$$

Ainsi l'opérateur de Laplace peut être approché de la manière suivante:

$$s \;\rightleftharpoons\; \frac{1}{T}\cdot(1-z^{-1}) \qquad (A.4)$$

A.2 Transformation d'une PDE en un système d'ODE

Considérons le cas d'un barreau élastique dans lequel se propage une perturbation (longitudinale) le long de l'axe X (fig. A- 1).

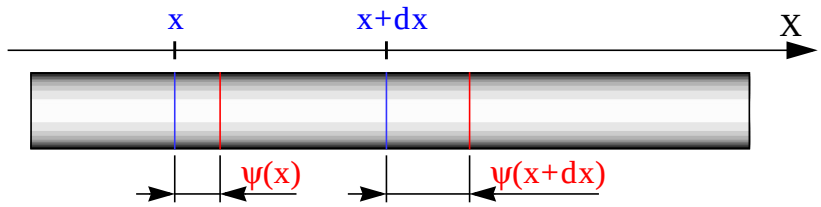


Figure A-1. Barreau élastique de section S, dans lequel se propage une perturbation longitudinale ψ.

En admettant que la perturbation se propage uniquement le long de l'axe X, le comportement de celle-ci est décrit par l'équation d'onde (A.5). En introduisant le module d'élasticité E, la densité ρ et la tension mécanique σ dans l'équation d'onde (A.5), nous obtenons l'équation (A.6). En reliant la contrainte mécanique σ à la force F et à la section du barreau S, nous obtenons l'équation (A.7).

$$\frac{\partial^2 \psi}{\partial t^2} \cdot \frac{1}{v^2} = \frac{\partial^2 \psi}{\partial x^2} \qquad (A.5)$$

$$\frac{\partial^2 \psi}{\partial t^2} \cdot \rho = \frac{\partial \sigma}{\partial x} \qquad (A.6)$$

avec $v = \sqrt{\frac{E}{\rho}}$ et $\sigma = E \cdot \frac{\partial \psi}{\partial x}$

$$S \cdot dx \cdot \rho \cdot \frac{\partial^2 \psi}{\partial t^2} = dF \qquad (A.7)$$

avec $\sigma = \frac{F}{S}$

masse · accélération = somme des forces

L'équation (A.7) correspond à la loi du mouvement de Newton (masse multipliant l'accélération est égale à la somme des forces).

En appliquant le concept des éléments finis, nous pouvons diviser le barreau en n éléments (fig. A- 2 a). A chaque élément sont attribuées une rigidité et une masse (fig. A- 2 b).

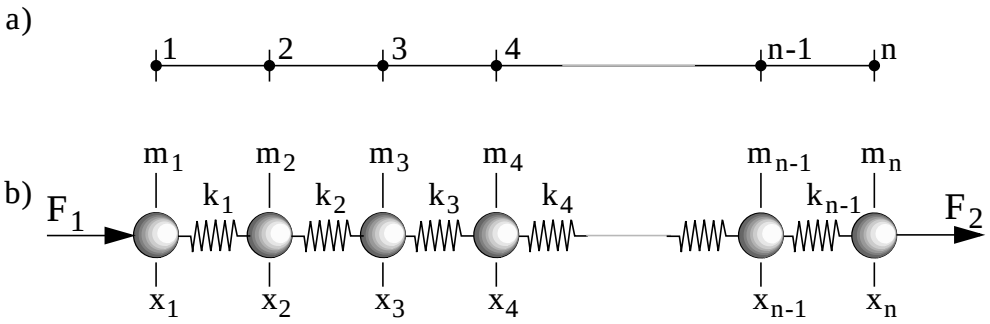


Figure A-2. (a) Décomposition en éléments finis, (b) auxquels on attribue des rigidités et des masses.

En appliquant la loi du mouvement à chacune des masses, nous obtenons le système d'ODE (A.8).

$$\mathbf{M} \cdot \ddot{\vec{x}} = \mathbf{K} \cdot \vec{x} + \vec{F} \qquad \& \qquad \ddot{\vec{x}} = \frac{d^2}{dt^2}\vec{x} \qquad \text{(A.8)}$$

où M est une matrice de masses diagonale de dimension n x n et K une matrice de rigidités, également de dimension n x n:

$$\mathbf{M} = \begin{bmatrix} m_1 & 0 & \dots & 0 \\ 0 & m_2 & & 0 \\ \dots & & & \dots \\ 0 & 0 & \dots & m_n \end{bmatrix} \qquad \vec{F} = \begin{bmatrix} F_1 \\ 0 \\ \dots \\ 0 \\ F_2 \end{bmatrix} \qquad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$

$$\mathbf{K} = \begin{bmatrix} -k_1 & k_1 & 0 & \dots & 0 \\ k_1 & -(k_1 + k_2) & k_2 & & 0 \\ 0 & k_2 & -(k_2 + k_3) & & 0 \\ \dots & & & & \dots \\ 0 & 0 & 0 & \dots & -k_{n-1} \end{bmatrix} \qquad \text{(A.9)}$$

Considérons le schéma électrique formé de capacités et d'inductances de la figure 3.

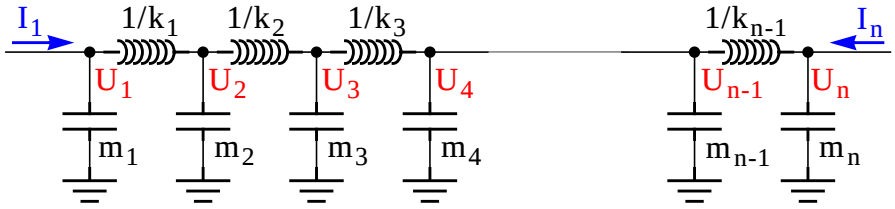


Figure A-3. Schéma électrique correspondant à la décomposition FE de la figure 1 b. U_i sont les tensions des n noeuds, I_1 et I_n sont les courants injectés aux noeuds 1 et n.

Dans le domaine de Laplace, les équations reliant les courants et tensions du schéma de la figure 3 sont (A.10).

$$\begin{cases} I_1 = s \cdot U_1 \cdot m_1 + \frac{1}{s} \cdot [U_1 - U_2] \cdot k_1 \\ I_2 = s \cdot U_2 \cdot m_2 - \frac{1}{s} \cdot [U_1 - U_2] \cdot k_1 + \frac{1}{s} \cdot [U_2 - U_3] \cdot k_2 \\ \quad \ldots \\ I_{n-1} = s \cdot U_{n-1} \cdot m_{n-1} - \frac{1}{s} \cdot [U_{n-2} - U_{n-1}] \cdot k_{n-2} \\ \qquad\qquad + \frac{1}{s} \cdot [U_{n-1} - U_n] \cdot k_{n-1} \\ I_n = s \cdot U_n \cdot m_n - \frac{1}{s} \cdot [U_{n-1} - U_n] \cdot k_{n-1} \end{cases} \qquad (A.10)$$

où les courants I_i et tensions U_i sont des fonctions de l'opérateur de Laplace s: $I_i(s)$ et $U_i(s)$.

En multipliant toutes les équations par l'opérateur de Laplace, puis en passant dans le domaine temporel tout en imposant des conditions initiales nulles, nous obtenons sous la forme matricielle le système d'équations différentielles ordinaires suivant:

$$M \cdot \ddot{\vec{U}} = K \cdot \vec{U} + \dot{\vec{I}} \qquad (A.11)$$

où M et K sont exactement les mêmes matrices que dans l'expression précédente (A.9).

A.3 Interpolation polynômiale

A.3.1 Introduction

Définitions

1. Soit f une fonction définie sur l'intervalle $[a, b]$, alors on dit que f est *injective* si f ne prend qu'une fois ses valeurs sur l'intervalle $[a, b]$. Formellement:

$$\text{f est injective} \Leftrightarrow \begin{cases} f(x_1) = f(x_2) \text{ pour } x_1 = x_2 \\ \text{si } x_1 \neq x_2 \text{ alors } f(x_1) \neq f(x_2) \end{cases} \qquad (A.12)$$

2. Soit S une région du plan xy. $\zeta^n(S)$ dénote la classe de fonctions continues et ayant les dérivées d'ordre 1,2,...,n continues sur S.

3. $\zeta^\infty(S)$ dénote la classe de fonctions continues et ayant toutes les dérivées continues sur S.

4. On appelle *polynôme* la fonction suivante:

$$p(x) = \sum_{i=0}^{n} a_i x^i = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x^1 + a_0 \qquad (A.13)$$

   où n est un nombre entier positif. Les coefficients a_i sont des nombres réels ou complexes.

5. Si $a_n \neq 0$ alors p(x) est de degré n.

6. Si tous les coefficients a_i égalent zéro, p(x) est appelé *polynôme zéro.*

7. $\boldsymbol{P_n}$ dénote la classe de tous les polynômes de degré plus petit ou égal à n, inclu le polynôme zéro.

Théorème

L'espace vectoriel $\boldsymbol{P_n}$ est de dimension n+1.

Remarques

1. Pour évaluer un polynôme p de degré n en un point x, il suffit de multiplier et d'additionner un nombre fini de fois des nombres réels.
2. Plus précisément, l'évaluation du polynôme (A.13) demande l'évaluation de $n(n+1)$ multiplications et n additions.
3. En revanche, le schéma de Horner ne requiert que n multiplications et n additions. Celui-ci est obtenu en introduisant des parenthèses dans l'expression (A.13):

$$p(x) = (\ldots(a_n \cdot x + a_{n-1}) \cdot x + \ldots + a_1) \cdot x + a_0 \qquad (A.14)$$

A.3.2 Interpolation polynômiale de courbes

Théorème

Si $\{x_0,x_1,\ldots,x_n\}$ est une collection de n+1 nombres distincts, alors, pour n'importe quelle collection de nombres $\{f_0,f_1,\ldots,f_n\}$, il existe un unique polynôme p(x) de la classe $\boldsymbol{P_n}$ satisfaisant $p(x_i)=f_i$ avec i=0,1,...,n.

Remarque

Lorsque n augmente, l'approximation semble s'améliorer au centre de l'intervalle. Inversement des effets d'oscillations sont observés aux extrémités de l'intervalle. Cela indique qu'un degré élevé ne signifie pas forcément que l'interpolation est bonne.

Calcul des coefficients

Pour calculer les coefficients a_i du polynôme p(x), nous avons n+1 conditions à respecter. Il s'agit des relations $p(x_i)=f_i$ avec i=0,1,...,n. Cela donne sous forme matricielle:

$$V \cdot \vec{a} = \vec{f} \qquad (A.15)$$

&

$$V = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ & & \dots & & \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix} \qquad \vec{a} = \begin{bmatrix} a_0 \\ a_1 \\ \dots \\ a_n \end{bmatrix} \qquad \vec{f} = \begin{bmatrix} f_0 \\ f_1 \\ \dots \\ f_n \end{bmatrix}$$

Le fait que les abscisses de la matrice sont toutes différentes indique que la matrice inverse existe et qu'il n'y a qu'une solution à ces équations:

$$\vec{a} = V^{-1} \cdot \vec{f} \qquad (A.16)$$

Formule d'interpolation de Newton

Ici, la différence entre les valeurs consécutives x_i de la variable indépendante x est supposée constante et égale à h. En introduisant la notation suivante que l'on appelle *différence du n-ème ordre.*

$$\begin{aligned} \Delta f_0 &= f_1 - f_0 & \Delta f_1 &= f_2 - f_1 & \dots \\ \Delta^2 f_0 &= \Delta f_1 - \Delta f_0 & \Delta^2 f_1 &= \Delta f_2 - \Delta f_1 & \dots \\ &\dots & &\dots & \\ \Delta^n f_0 &= \Delta^{n-1} f_1 - \Delta^{n-1} f_0 & & & \end{aligned} \qquad (A.17)$$

Nous pouvons écrire les polynômes prenant respectivement les valeurs des premiers aux derniers points. Il s'agit de la *formule d'interpolation de Newton*:

$$p_1(x) = f_0 + \Delta f_0 \cdot \frac{x - x_0}{h}$$

$$p_2(x) = p_1(x) + \frac{\Delta^2 f_0}{2!} \cdot \frac{x - x_0}{h} \cdot \left(\frac{x - x_0}{h} - 1 \right) \qquad \text{(A.18)}$$

$$\ldots$$

$$p_n(x) = p_{n-1}(x) + \frac{\Delta^n f_0}{n!} \cdot \frac{x - x_0}{h} \cdot \left(\frac{x - x_0}{h} - 1 \right) \cdot \ldots \cdot \left[\frac{x - x_0}{h} - (n - 1) \right]$$

La particularité de ce type de polynômes réside dans le fait qu'en passant au polynôme de degré supérieur les premiers termes ne sont pas modifiés. Seul un nouveau terme vient s'ajouter.

Formule d'interpolation de Lagrange

L'idée est de résoudre le problème d'interpolation en prenant les x_i, mais en ne considérant la collection de f_i qu'à tour de rôle. Selon le théorème, la solution à ce problème n'a qu'une seule solution. Cette solution est dénotée par les fonctions suivantes qui sont appelées les *polynômes fondamentaux de Lagrange.*

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} \qquad \& \qquad L_i(x_j) = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases} \qquad \text{(A.19)}$$

La solution s'exprime alors de la manière suivante: il s'agit de la *formule d'interpolation de Lagrange* pour n+1 points:

$$p_n(x) = \sum_{i=0}^{n} f_i \cdot L_i(x) \qquad \text{(A.20)}$$

Il est possible de démontrer que chaque polynôme de $\boldsymbol{P_n}$ peut être mis sous la forme de cette dernière équation. En introduisant les coefficients a_n, il est également possible de réécrire la formule d'interpolation de Lagrange de la manière suivante:

$$a_n = \frac{f_n - p_{n-1}(x_n)}{f_0} \cdot \prod_{i=1}^{n-1} \frac{y_0}{p_i(x_n) - y_i}$$

$$p_n(x) = p_{n-1}(x) + a_n \cdot y_0 \cdot \prod_{i=1}^{n-1} \frac{p_i(x) - y_i}{y_0} \qquad (A.21)$$

L'avantage de cette notation réside, comme pour la formule d'interpolation de Newton, dans le fait qu'en passant au polynôme de degré supérieur les premiers termes ne sont pas modifiés.

Méthode des moindres carrés

Admettons maintenant que le nombre de points de mesures est n+1 et que nous cherchons à les interpoler avec un polynôme de la classe $\boldsymbol{P_m}$, où $m \leq n$. La différence entre le polynôme et les points de mesures $p(x_i)$-f_i est appelée *déviation*. L'idée est de minimiser la somme des déviations au carré:

$$E[p] = \sum_{i=0}^{n} [p(x_i) - f_i]^2 \qquad (A.22)$$

Il s'agit de la méthode également appelée *méthode des moindres carrés*. Les conditions nécessaires pour obtenir un minimum sont d'annuler les dérivées partielles suivantes:

$$\frac{\partial E}{\partial a_i} = 0 \quad \& \quad i = 0, 1, \ldots, m \qquad (A.23)$$

Sous forme matricielle, cela revient à écrire l'expression suivante:

$$V^T \cdot V \cdot \vec{a} = V^T \cdot \vec{f} \qquad (A.24)$$

&

$$V = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^m \\ 1 & x_1 & x_1^2 & \dots & x_1^m \\ & & \dots & & \\ 1 & x_n & x_n^2 & \dots & x_n^m \end{bmatrix} \qquad \vec{a} = \begin{bmatrix} a_0 \\ a_1 \\ \dots \\ a_m \end{bmatrix} \qquad \vec{f} = \begin{bmatrix} f_0 \\ f_1 \\ \dots \\ f_n \end{bmatrix}$$

Lorsque $m = n$, le nombre de coefficients est égal au nombre de points de mesures et la somme des déviations au carré E[p] vaut zéro.

Théorème

Si les x_i sont distincts et que $m \leq n$, alors il existe un unique polynôme pour lequel la somme des déviations au carré est minimum. De plus les coefficients du polynôme sont donnés par la solution du système d'équation (A.24).

Remarques

1. Une variante de la méthode des moindres carrés consiste à introduire une fonction "poids" $w_i(x)$ dans la somme des déviations au carré.

$$\sum_{i=0}^{n} w_i(x) \cdot [p(x_i) - f_i]^2 \qquad (A.25)$$

   Cette fonction permet d'accorder plus ou moins de poids à certaines mesures. Mais, pour obtenir un minimum, l'effort de calcul est important. C'est pour cette raison que cette variante n'est à utiliser que lorsqu'elle est vraiment nécessaire.

2. Il est également envisageable de minimiser autre chose que la somme des déviations au carré. Par exemple, la somme des valeurs absolues des déviations (A.26) ou bien la valeur maximum des valeurs absolues des déviations (A.27). Mais ces deux exemples donnent des équations non-linéaires plus difficiles à résoudre. C'est pour cette raison que c'est généralement la méthode des moindres carrés qui est utilisée.

$$\sum_{i=0}^{n} |p(x_i) - f_i| \qquad (A.26)$$

$$\max_{0 \le i \le n} |p(x_i) - f_i| \qquad (A.27)$$

A.3.3 Interpolation polynômiale de surfaces

Dans ce chapitre, nous nous intéressons aux polynômes composés de deux variables indépendantes x et y. Comme pour les polynômes d'une variable indépendante, il est possible de définir des classes de polynômes $\boldsymbol{P_n}$. La table suivante énumère les monômes des quatre premières classes.

Classes	*Monômes*
$\boldsymbol{P_0}$	1
$\boldsymbol{P_1}$	1, x, y
$\boldsymbol{P_2}$	1, x, y, x^2, xy, y^2
$\boldsymbol{P_3}$	1, x, y, x^2, xy, y^2, x^3, x^2y, xy^2, y^3

Table B-1. Classes de polynômes et énumérations des monômes.

Ainsi un polynôme de la classe $\boldsymbol{P_n}$ contient des monômes de la forme $x^i y^i$, où $0 \le i + j \le n$ et $0 \le i$, $0 \le j$. Remarquons que l'espace vectoriel $\boldsymbol{P_n}$ est de dimension:

$$(n+1)(n+2)/2 \qquad (A.28)$$

Les polynômes composés de deux variables indépendantes définissent une surface dans l'espace à trois dimensions. Il s'agit de surfaces similaires à ce que nous avons vu au chapitre 4 (cf. fig. 4-4 p. 66 et fig. 4-7 p. 72).

Remarque

Dans le cas d'un polynôme à une variable indépendante, nous avons vu que si nous respectons certaines conditions, il existe un théorème nous assurant de trouver une solution. Dans le cas de deux variables indépendantes, il n'existe pas un tel théorème. Cela signifie qu'en général, pour un choix arbitraire de mesures, il n'existe pas de polynôme permettant leur interpolation exacte. C'est-à-dire tel que $p(x_i,y_i)=f_i$.

Méthode des moindres carrés

De façon similaire au cas à une dimension, admettons que le nombre de points de mesures est n+1. C'est-à-dire que nous avons les valeurs f_i pour les points x_i et y_i (i=0, 1, ..., n). Admettons encore, afin d'illustrer l'approche, que l'on désire interpoler ces points avec une surface quadratique. C'est-à-dire un polynôme appartenant à la classe $\boldsymbol{P_2}$ tel que:

$$p(x,y) = a_1 + a_2x + a_3y + a_4x^2 + a_5xy + a_6y^2 \qquad (A.29)$$

Comme pour le cas à une dimension, nous devons minimiser la somme des déviations au carré.

$$E[p] = \sum_{i=0}^{n} [p(x_i, y_i) - f_i]^2 \qquad (A.30)$$

Comme pour le cas à une dimension, les conditions nécessaires pour y parvenir sont:

$$\frac{\partial E}{\partial a_i} = 0 \qquad \& \qquad i = 0, 1, \ldots, n \qquad (A.31)$$

Sous forme matricielle, nous obtenons le système d'équations suivantes:

$$V^T \cdot V \cdot \vec{a} = V^T \cdot \vec{f} \qquad (A.32)$$

&

$$V = \begin{bmatrix} 1 & x_0 & y_0 & x_0^2 & x_0 y_0 & y_0^2 \\ 1 & x_1 & y_1 & x_1^2 & x_1 y_1 & y_1^2 \\ & & & \ldots & & \\ 1 & x_n & y_n & x_n^2 & x_n y_n & y_n^2 \end{bmatrix} \qquad \vec{a} = \begin{bmatrix} a_0 \\ a_1 \\ \ldots \\ a_6 \end{bmatrix} \qquad \vec{f} = \begin{bmatrix} f_0 \\ f_1 \\ \ldots \\ f_n \end{bmatrix}$$

Pour que ce système ait une solution, il faut que les points de mesures soient distincts et que $n + 1 \geq 6$.

Annexes B

GLOBS Design-Kit

B.1 Outils utilisés

L'environnement de CAO que nous avons choisi pour réaliser des simulations globales de microsystèmes est un environnement développé pour la CAO d'électronique numérique et analogique fonctionnant sur station Sun SPARC sous le système d'exploitation Solaris 2.5 (UNIX). Il s'agit de l'environnement de Mentor Graphics® (c.f. http://www.mentorg.com). Le principal avantage de cet environnement est qu'il possède plusieurs simulateurs. Ceux-ci autorisent la simulation de modèles analogiques tels que SPICE ou HDL-A, et numériques tels que VHDL. Pour utiliser le design-kit GLOBS, il faut disposer des outils suivants (fig. B- 1 & fig. B- 2):

1. Design Architect (*DA*) est un éditeur de schéma utilisé pour entrer et assembler des parties de design sous la forme de schémas.
2. Accusim II HDL-A est un simulateur de temps continu analogique utilisé pour simuler des modèles de type SPICE et HDL-A.
3. QuickHDL est un simulateur numérique utilisé pour simuler des modèles VHDL.
4. QuickSim II est un simulateur numérique utilisé pour simuler des modèles numériques de portes logiques (gate-level).

5. QuickHDL Pro réunit les simulateurs QuickHDL et QuickSim II. Il permet donc la réalisation de simulations numériques de modèles VHDL branchés à des portes logiques.

6. Continuum-QuickHDL réunit les simulateurs QuickHDL, QuickSim II et Accusim II HDL-A. Cet outil permet donc la réalisation de simulations globales.

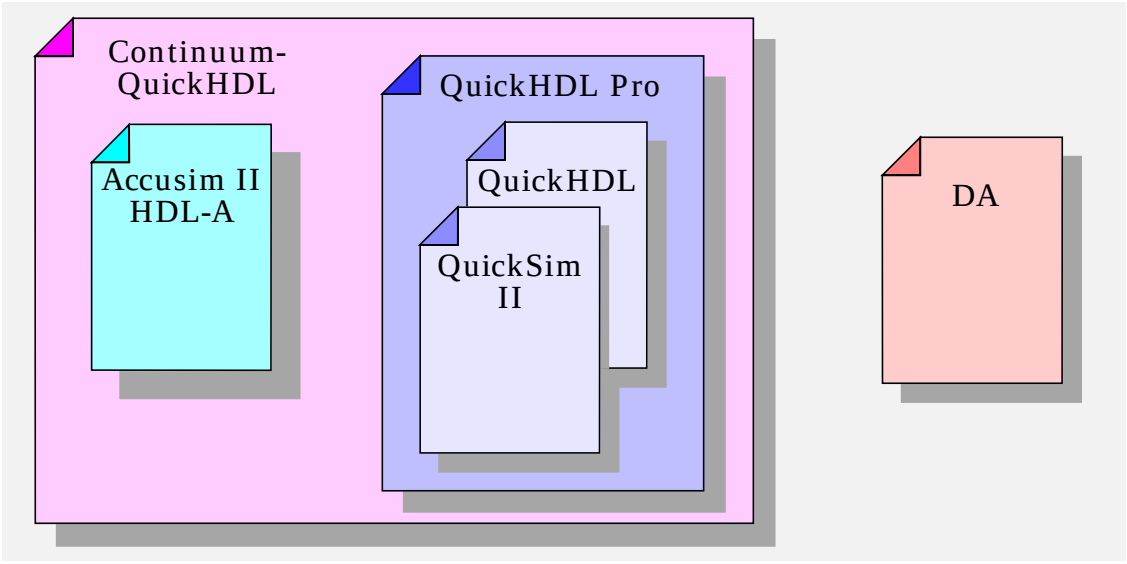


Figure B-1. Environnement de Mentor Graphics® version B.3.

Pour effectuer de la co-simulation hardware-software, il faut disposer de QuickHDL, de Tcl/Tk et de l'environnement de développement de MicroChips nommé MPLAB™ (fig. B- 2).

7. Tcl/Tk 8.0 est utilisable gratuitement et peut être téléchargé chez Sun Microsystems Laboratories (http://www.sunscript.sun.com).

8. MPLAB 3.22.00 est un environnement de développement de code pour microcontrôleurs PIC. Cet environnement fonctionne sur PC sous le système d'exploitation Windows. MPLAB est également utilisable gratuitement et peut être téléchargé chez Microchip Technology Inc. (http://www.microchip.com).

Les traducteurs (c.f. fig. B- 2) sont des programmes UNIX fournis par le design-kit GLOBS. Les flèches nommées IPC (UNIX Inter Process Communication) représentent les canaux de communica-

tion entre le simulateur VHDL (QuickHDL) et la fenêtre Tcl/Tk. Celle-ci fait également partie du design-kit GLOBS.

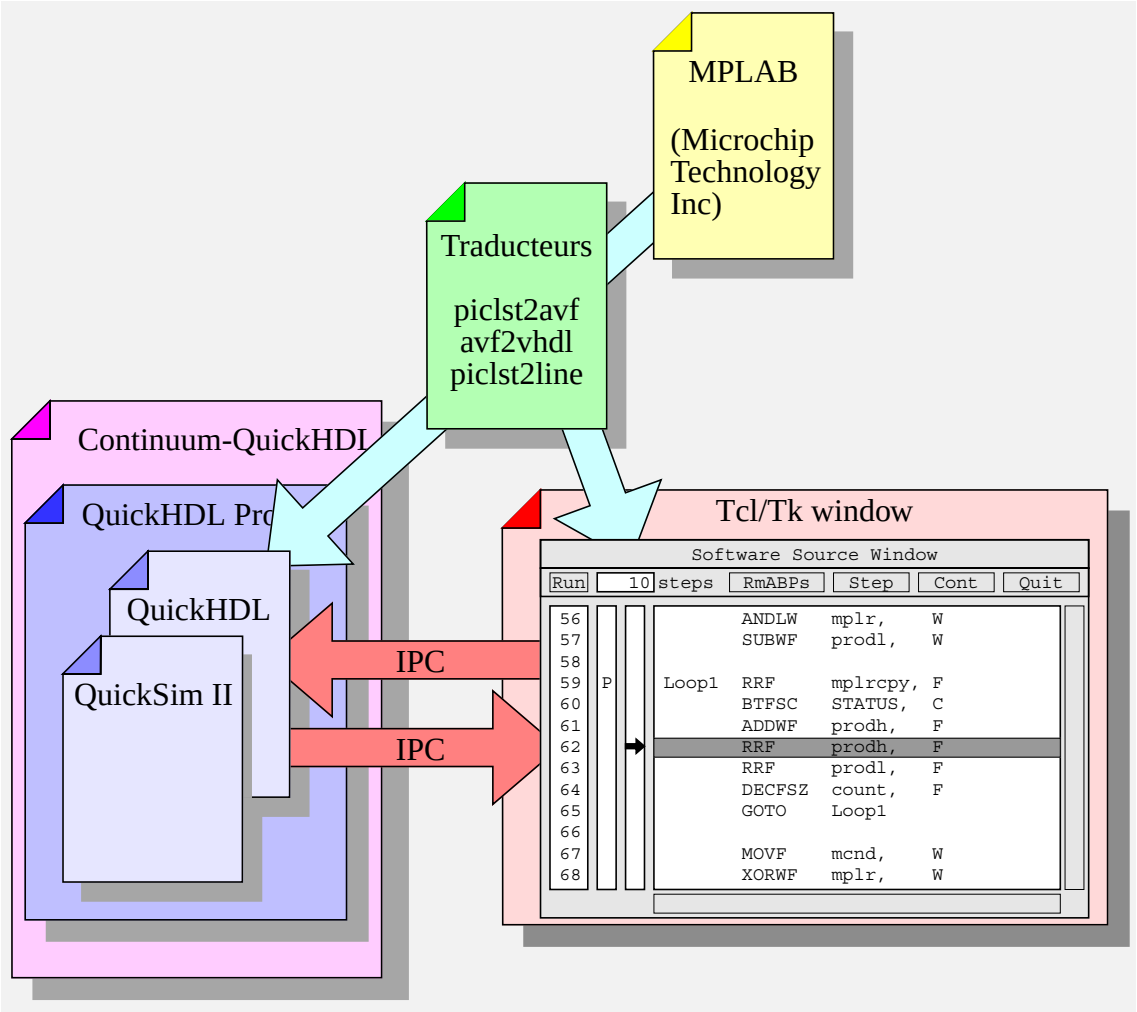


Figure B-2. Co-simulation hardware-software, outils utilisés.

B.2 Bibliothèque de modèles

La table suivante énumère les modèles se trouvant dans le design-kit GLOBS. Dans cette table, on trouve également le type (VHDL, HDL-A, etc.) ainsi qu'une description de chaque modèle. Certains des modèles sont utilisés pour composer d'autres modèles. Ainsi le modèle du microrelais utilise quatre autres modèles (contact1, magstop1, magtrans1 et oscillator1). Pour effectuer de la co-simulation hardware-software, il faut également utiliser quatre modèles (addr2line, hsw_tkintf, pic16c5xiss_v2 et picrom). Remarquons pour finir, que les modèles de sources de courant et de tension sont

extrêmement simples mais indispensables pour créer des bancs de tests. Pour plus de détails, veuillez consulter le manuel se trouvant sur le CD du design-kit GLOBS.

Modèle	Type	Description succincte
addr2line	VHDL	Donne le numéro de la ligne de programme correspondant à l'adresse pointée par un microcontrôleur de la famille PIC16C5X.
ceil1	HDL-A/C	Arrondit la grandeur d'entrée across vers l'entier le plus proche dans la direction de l'infini.
clk1	HDL-A	Génère un signal numérique d'horloge.
clk2	VHDL	Génère un signal numérique d'horloge.
contact1	HDL-A	Modèle comportemental d'un bras de contact utilisé dans la modélisation d'un micro-relais.
dampedmass1	SPICE	Modèle comportemental d'une masse amortie.
exp1	SPICE	Source de tension ou de courant d'une impulsion de type exponentielle amortie.
floatq1	HDL-A/C	Quantifie la grandeur d'entrée across en utilisant la représentation à virgule flottante normalisée IEEE754.
floor1	HDL-A/C	Arrondit la grandeur d'entrée ACROSS vers l'entier le plus proche dans la direction de moins l'infini.
friction1	HDL-A	Modèle comportemental d'une masse en contact avec une surface.
friction2	HDL-A	Modèle comportemental d'une masse reliée par un ressort à une autre masse en contact avec une surface.
friction3	HDL-A	Modèle comportemental d'une masse reliée par un ressort à une plaque sans masse en contact avec une surface.

Table B-1. Liste et descriptions succinctes des modèles se trouvant dans le design-kit GLOBS.

Modèle	Type	Description succincte
friction4	HDL-A	Modèle comportemental d'une masse reliée par une matrice de rigidité à une plaque sans masse en contact avec une surface.
hsw_tkintf	VHDL/ C	Canal de communication utilisé pour la co-simulation hardware-software.
id_opamp1	SPICE	Macro-modèle d'un amplificateur opérationnel idéal.
imt1614_v1	SPICE/ HDL-A	Modèle comportemental du convertisseur AN/NA IMT 1614. Il s'agit d'un convertisseur sériel dont la dynamique est de 14 bits.
iv_transformer1	SPICE	Transforme une grandeur through en une grandeur across.
iv_transformer2	HDL-A	Transforme une grandeur through en une grandeur across.
iv_transformer3	SPICE	Transforme une grandeur through en une grandeur across.
magstop1	HDL-A	Modèle comportemental d'une butée utilisée dans la modélisation d'un micro-relais.
magtrans1	HDL-A	Modèle comportemental d'un capteur/ actionneur électromagnétique.
microrelay1	SPICE/ HDL-A	Modèle comportemental d'un micro-relais.
one_pulse1	HDL-A	Génère une impulsion périodique.
oscillator1	SPICE	Modèle comportemental d'un oscillateur harmonique amorti.
pattern1	SPICE	Source de courant ou de tension d'impulsions.
pic16c5xiss_v2	VHDL	Instruction Set Simulator (ISS) d'un microcontrôleur de la famille PIC16C5X.
picrom	VHDL	ROM d'un microcontrôleur de la famille PIC16C5X.

Table B-1. Liste et descriptions succinctes des modèles se trouvant dans le design-kit GLOBS.

Modèle	Type	Description succincte
pid1	HDL-A	Modèle comportemental d'un régulateur de type PID.
pid2	SPICE	Modèle comportemental d'un régulateur de type PID.
piezo1	HDL-A	Modèle comportemental d'un capteur/ actionneur piézoélectrique.
piezo2	SPICE	Modèle comportemental d'un capteur/ actionneur piézoélectrique.
piezo3	HDL-A	Modèle comportemental d'un capteur/ actionneur piézoélectrique.
piezo4	SPICE	Modèle comportemental d'un capteur/ actionneur piézoélectrique.
piezores1	HDL-A	Modèle comportemental d'une piézo-résistance diffusée dans une membrane de silicium. La valeur de la résistance piézo-résistance dépend de la température et de la pression appliquée à la membrane.
pow1	HDL-A/ C	Réalise l'opération mathématique "puissance de n" de la grandeur across d'entrée.
pulse1	SPICE	Source de courant ou de tension d'une impulsion périodique.
pwl1	SPICE	Source de courant ou de tension d'une fonction linéaire par morceaux.
reset1	VHDL	Génère un signal d'initialisation.
rod1	SPICE	Modèle comportemental d'un barreau n'ayant qu'une dimension.
round1	HDL-A/ C	Arrondit la grandeur d'entrée across vers l'entier le plus proche.
rquant1	HDL-A/ C	Quantification de type complément à deux de la grandeur d'entrée across.
sffm1	SPICE	Source de courant ou de tension d'un sinus modulé en fréquence.

Table B-1. Liste et descriptions succinctes des modèles se trouvant dans le design-kit GLOBS.

Modèle	Type	Description succincte
sigdelmod1	HDL-A	Modèle comportemental d'un modulateur delta-sigma.
sin1	SPICE	Source de courant ou de tension d'un sinus ou d'un sinus amorti.
transformer1	HDL-A	Modèle comportemental d'un transformateur électrique idéal.
twoscq1	HDL-A/C	Quantification de type complément à deux de la grandeur d'entrée across.
xva_measure1	HDL-A	Transforme l'entrée across représentant la vitesse, en déplacement et accélération correspondante.

Table B-1. Liste et descriptions succinctes des modèles se trouvant dans le design-kit GLOBS.

Annexes C

Exemples de modèles

C.1 Modèle HDL-A d'une diode à jonction

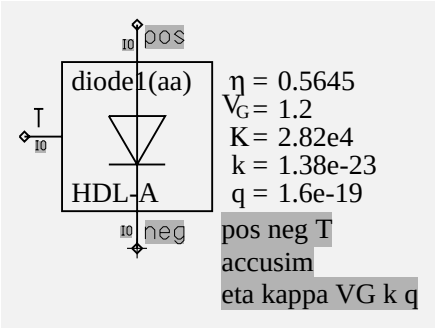


Figure B-3. Symbole du modèle de la diode dans Design Architect.

```
ENTITY diode IS
   GENERIC (eta, kappa, VG, k, q   : real);
   PIN     (pos, neg               : electrical;
            T                      : thermal);
END ENTITY diode;

ARCHITECTURE aa OF diode IS
  VARIABLE ISS,UD,Temp,ID  : real;
BEGIN
  RELATION
  PROCEDURAL FOR ac, dc, transient =>
    UD  := real([pos,neg].v);
    Temp:= real(T.t)+273.15;
    ISS := kappa*(Temp**eta)*exp(-q*VG/(k*Temp));
    ID  := ISS*(exp(q*UD/(k*Temp))-1.0);
    [pos,neg].i %= ID;
```

```
  END RELATION;
END ARCHITECTURE aa;

ARCHITECTURE bb OF diode IS
  VARIABLE ISS,UD,Temp,ID  : real;
BEGIN

  RELATION
  PROCEDURAL FOR ac, dc, transient =>
    ID  := real(pos.i);
    Temp:= real(T.t)+273.15;
    ISS := kappa*(Temp**eta)*exp(-q*VG/(k*Temp));
    UD  := ln((ID+ISS)/ISS)*k*Temp/q;
    [pos,neg].v %= UD;
  END RELATION;
END ARCHITECTURE bb;
```

C.2 Modèle SPICE d'une diode à jonction

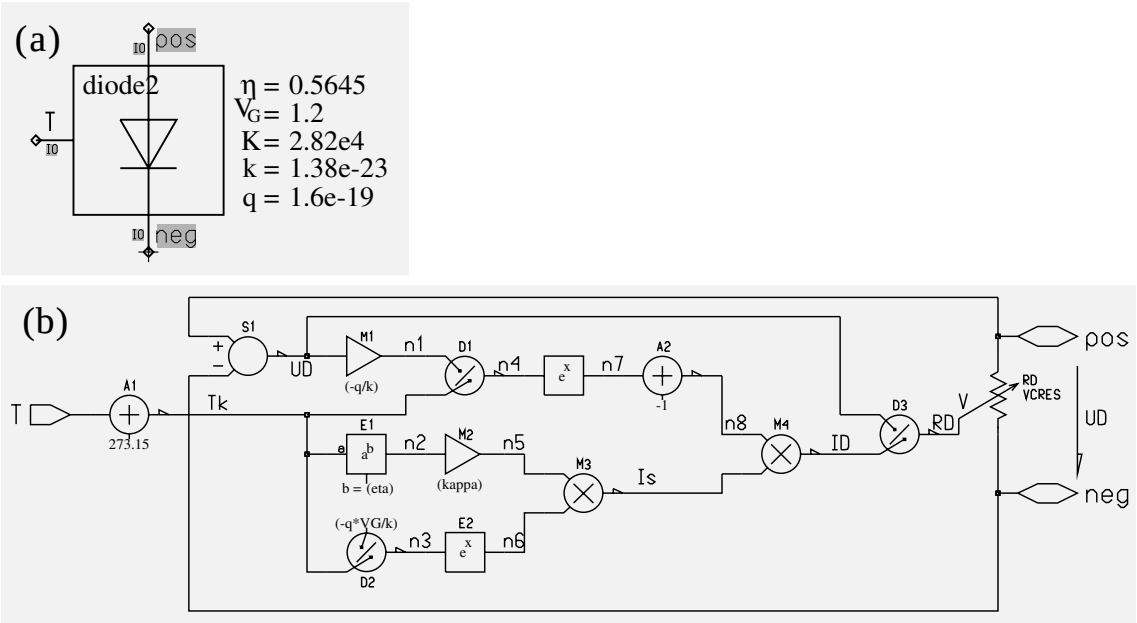


Figure B-4. (a) Symbole et (b) schéma SPICE du modèle de la diode dans Design Architect.

```
.subckt diode2 pos neg T
V_A1_C1 A1_n1 0 273.15
V_D2_C1 D2_n1 0 DC -13913
V_M2_C1 M2_n1 0 DC 1
V_E1_C1 E1_n1 0 0.5645
V_A2_C1 A2_n1 0 -1
V_M1_C1 M1_n1 0 DC 1
X_RD pos neg RD  VCRES  k=1
```

```
X_A1_A1 T A1_n1 Tk  GEN_ADD  k1=1 k2=1
X_A2_A1 n7 A2_n1 n8  GEN_ADD  k1=1 k2=1
X_D3 UD ID RD  DIVIDE  k=1
X_D2_D1 D2_n1 Tk n3  DIVIDE  k=1
X_D1 n1 Tk n4  DIVIDE  k=1
X_M4 n8 Is ID  MULTIPLY  k=1
X_M3 n5 n6 Is  MULTIPLY  k=1
X_M2_M1 n2 M2_n1 n5  MULTIPLY  k=28200
X_E1_E1_M1 E1_E1_lna E1_n1 E1_E1_blna  MULTIPLY  k=1
X_M1_M1 UD M1_n1 n1  MULTIPLY  k=-11594.2
X_E2 n3 n6  EXP
X_E1_E1_E1 E1_E1_blna n2  EXP
X_I$3113 n4 n7  EXP
X_E1_E1_L1 Tk E1_E1_lna  LN
X_S1 pos neg UD  BAS_SUBTRACT  k1=1 k2=1
.ends diode2
```

C.3 Modèle VHDL-AMS d'une diode à jonction

```
LIBRARY disciplines;
USE disciplines.electromagnetic_system.all;
USE disciplines.thermal_system.all;
LIBRARY ieee;
USE ieee.math_real.all;

ENTITY diode IS
 GENERIC (eta, kappa, VG, k, q : real);
 PORT    (TERMINAL p,n          : electrical;
          TERMINAL Temp         : thermal);
END diode;

ARCHITECTURE aa OF diode IS
  QUANTITY diode_v ACROSS diode_i THROUGH p TO n;
  QUANTITY T ACROSS TP THROUGH Temp;
BEGIN
  TP == 0.0;
  diode_i == kappa*(T+273.15)**eta *
             exp(-q*VG/(k*(T+273.15))) *
             ( exp(diode_v*q/((T+273.15)*k)) - 1.0 );
END aa;

ARCHITECTURE bb OF diode IS
  QUANTITY diode_v ACROSS diode_i THROUGH p TO n;
  QUANTITY T ACROSS TP THROUGH Temp;
BEGIN
  TP == 0.0;
  diode_v == log2((diode_i+(kappa*((T+273.15)**eta)*
             exp(-q*VG/(k*(T+273.15))))))/
             (kappa*((T+273.15)**eta)*
             exp(-q*VG/(k*(T+273.15))))))*
             k*(T+273.15)/q;
END bb;
```

C.4 Modèle HDL-A d'une piézo-résistance

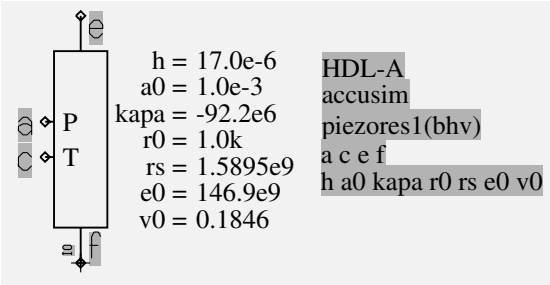


Figure B-5. Symbole du modèle HDL-A de la piézo-résistance dans Design Architect.

```
ENTITY piezores IS
  GENERIC (h, a0, kapa, r0, rs, e0, v0 : real);
  PIN     (a                           : fluid;
           c                           : thermal;
           e, f                        : electrical);
END ENTITY piezores;

ARCHITECTURE bhv OF piezores IS
  STATE     w11,AA,CC,GG,NTp,LTp,LTpt   : analog;
BEGIN
  RELATION
  PROCEDURAL FOR init =>
    AA  := h**3/(12.0*(1.0-v0**2));
    CC  := 0.2522;
    GG  := 0.000133*a0**4/h;
  PROCEDURAL FOR ac, dc, transient =>
    NTp := GG*a.p/(AA*(e0+kapa*c.t));
    LTp := ((2.7e6*NTp+sqrt(108.0*CC**3+(2.7e6*NTp)**2))/
           2.0);
    w11 := h*(LTpt/3.0-CC/LTpt);
    [e,f].i %= [e,f].v/(r0+(2.5223*rs*w11));
  EQUATION (LTpt) FOR ac,dc,transient =>
      LTp == LTpt**3;
  END RELATION;
END ARCHITECTURE bhv;
```

C.5 Modèle SPICE d'un capteur de pression piézo-résistif

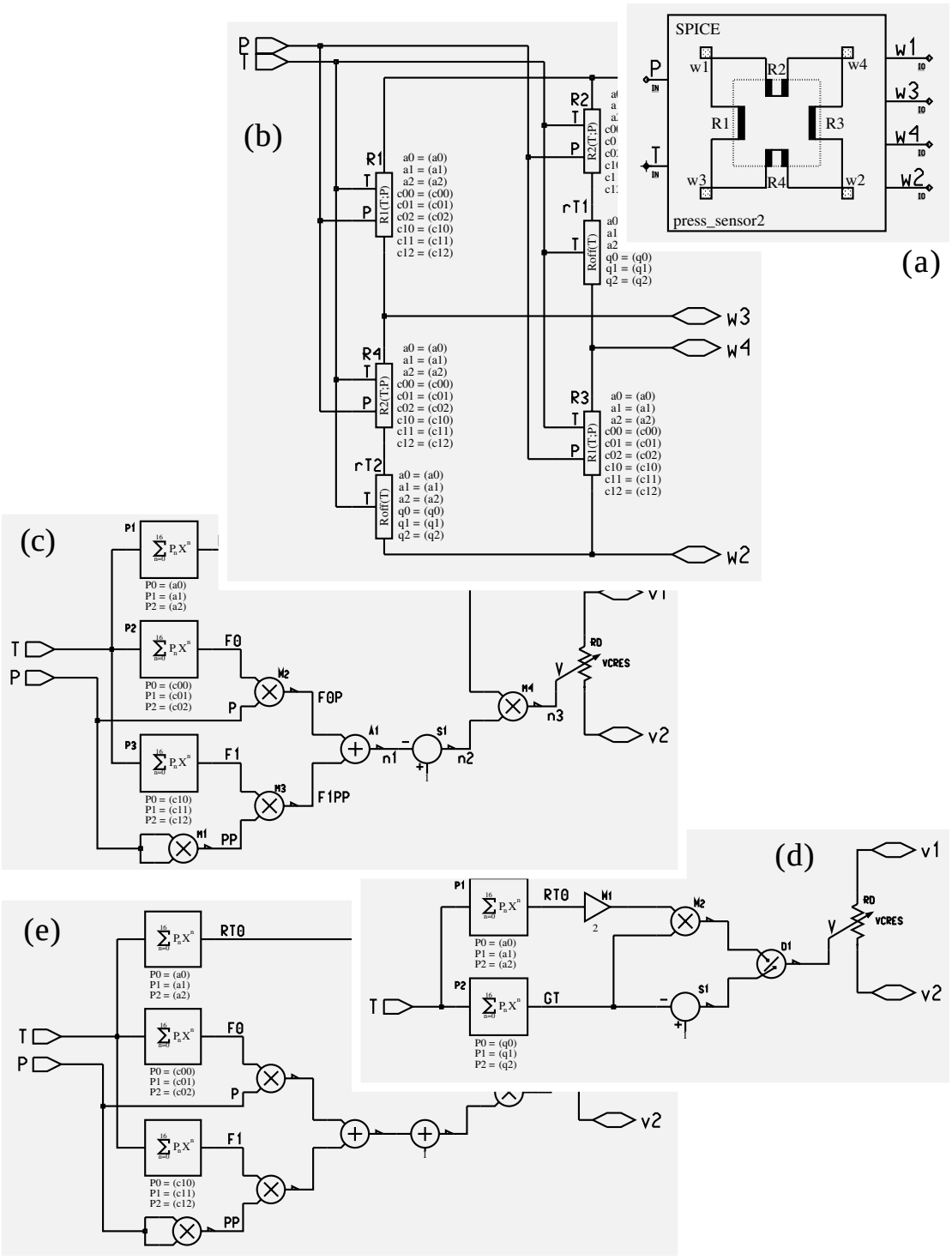


Figure B-6. (a) Symbole et (b) schéma du modèle SPICE du capteur de pression piézo-résistif dans Design Architect. Schémas des modèles de piézo-résistances de type (c) R1 = R3, (e) R2 = R4 et (d) rT1 = rT2.

```
.subckt R1 T P v1 v2
V_S1_C1 S1_n1 0 1
X_S1_S1 S1_n1 n1 n2  BAS_SUBTRACT  k1=1 k2=1
X_RD v1 v2 n3  VCRES  k=1
X_M4 RT0 n2 n3  MULTIPLY  k=1
X_M3 F1 PP F1PP  MULTIPLY  k=1
X_M2 F0 P F0P  MULTIPLY  k=1
X_M1 P P PP  MULTIPLY  k=1
X_A1 F0P F1PP n1  GEN_ADD  k1=1 k2=1
X_P3 T F1  POLY  p3=0 p4=0 p5=0 p6=0 p7=0 p8=0 p9=0
   p10=0 p11=0 p12=0 p13=0 p14=0 p15=0 p16=0
X_P2 T F0  POLY  p3=0 p4=0 p5=0 p6=0 p7=0 p8=0 p9=0
   p10=0 p11=0 p12=0 p13=0 p14=0 p15=0 p16=0
X_P1 T RT0  POLY  p3=0 p4=0 p5=0 p6=0 p7=0 p8=0 p9=0
   p10=0 p11=0 p12=0 p13=0 p14=0 p15=0 p16=0
.ends R1

.subckt R2 T P v1 v2
V_I$10_C1 I$10_n1 0 1
X_I$12 v1 v2 N$11  VCRES  k=1
X_I$11 RT0 N$10 N$11  MULTIPLY  k=1
X_I$8 F1 PP N$8  MULTIPLY  k=1
X_I$7 F0 P N$7  MULTIPLY  k=1
X_I$4 P P PP  MULTIPLY  k=1
X_I$10_A1 N$9 I$10_n1 N$10  GEN_ADD  k1=1 k2=1
X_I$9 N$7 N$8 N$9  GEN_ADD  k1=1 k2=1
X_I$3 T F1  POLY  p3=0 p4=0 p5=0 p6=0 p7=0 p8=0 p9=0
  p10=0 p11=0 p12=0 p13=0 p14=0 p15=0 p16=0
X_I$2 T F0  POLY  p3=0 p4=0 p5=0 p6=0 p7=0 p8=0 p9=0
  p10=0 p11=0 p12=0 p13=0 p14=0 p15=0 p16=0
X_I$1 T RT0  POLY  p3=0 p4=0 p5=0 p6=0 p7=0 p8=0 p9=0
  p10=0 p11=0 p12=0 p13=0 p14=0 p15=0 p16=0
.ends R2

.subckt rT T v1 v2
V_S1_C1 S1_n1 0 1
V_M1_C1 M1_n1 0 DC 1
X_RD v1 v2 N$7  VCRES  k=1
X_D1 N$4 N$6 N$7  DIVIDE  k=1
X_S1_S1 S1_n1 GT N$6  BAS_SUBTRACT  k1=1 k2=1
X_M2 N$1 GT N$4  MULTIPLY  k=1
X_M1_M1 RT0 M1_n1 N$1  MULTIPLY  k=2
X_P2 T GT  POLY  p3=0 p4=0 p5=0 p6=0 p7=0 p8=0 p9=0
  p10=0 p11=0 p12=0 p13=0 p14=0 p15=0 p16=0
X_P1 T RT0  POLY  p3=0 p4=0 p5=0 p6=0 p7=0 p8=0 p9=0
  p10=0 p11=0 p12=0 p13=0 p14=0 p15=0 p16=0
.ends rT
```

C.6 Modèle HDL-A d'un capteur de pression piézo-résistif

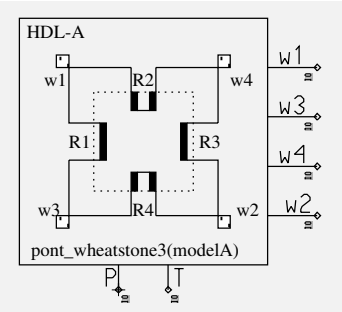


Figure B-7. Symbole du modèle HDL-A du capteur de pression piézo-résistif dans Design Architect.

```
ENTITY Pont_Wheatstone3 IS
  GENERIC (a0,a1,a2     : real;
           c00,c01,c02  : real;
           c10,c11,c12  : real;
           q0,q1,q2     : real);
  PIN     (w1,w2,w3,w4  : electrical;
           p            : fluid;
           T            : thermal);
END ENTITY Pont_Wheatstone3;

ARCHITECTURE modelA OF Pont_Wheatstone3 IS
  STATE   v1,v2,v3,v4,RT      : analog;
  STATE   F0T,F1T,GT,R1       : analog;
  STATE   R2,i1,i2,i3,i4,rrT  : analog;
BEGIN
  RELATION
  PROCEDURAL FOR init =>
    -- Resistance du pont
     a0  := 3551.88903773864;
     a1  := 9.56285906968;
     a2  := 0.01807821028;
    -- Constantes definissant le TCS
     c00 := 5.529385099e-2;
     c01 :=-1.147880908e-4;
     c02 := 1.23714792e-7;
     c10 :=-1.272959483e-3;
     c11 := 9.960596534e-6;
     c12 :=-1.7743922e-8;
    -- Constantes definissant l'offset
     q0  := 1.789790847e-3;
     q1  := 4.568186884e-6;
     q2  := 1.520472e-9;
  PROCEDURAL FOR ac,dc,transient =>
       RT   := a0+a1*T.T+a2*T.T*T.T;
       F0T  := c00+c01*T.T+c02*T.T*T.T;
       F1T  := c10+c11*T.T+c12*T.T*T.T;
       GT   := q0+q1*T.T+q2*T.T*T.T;
       rrT  := 2.0*RT*GT/(1.0-GT);
       R1   := RT*(1.0-(F0T*p.p+F1T*p.p*p.p));
```

```
        R2    := RT*(1.0+(F0T*p.p+F1T*p.p*p.p))+rrT;
        v1    := [w1,w3].v;
        v2    := [w1,w4].v;
        v3    := [w4,w2].v;
        v4    := [w3,w2].v;
        w1.i %= i1;
        w2.i %= i2;
        w3.i %= i3;
        w4.i %= i4;
  EQUATION (i1,i2,i3,i4) FOR dc,ac,transient =>
        i1 == v1/R1+v2/R2;
        i2 ==-v4/R2-v3/R1;
        i3 == v4/R2-v1/R1;
        i4 == v3/R1-v2/R2;
  END RELATION;
END ARCHITECTURE modelA;
```

C.7 Modèle SPICE d'un capt. action. piézoélectrique

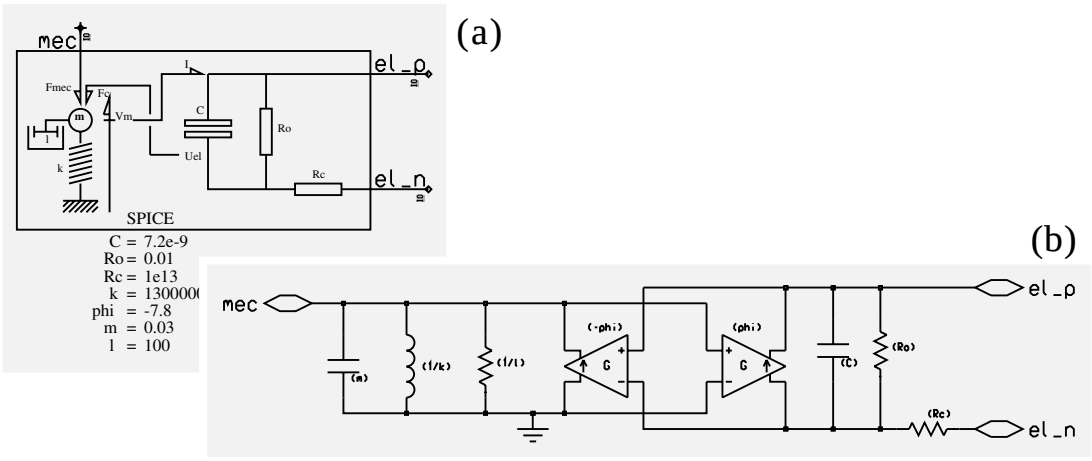


Figure B-8. (a) Symbole et (b) schéma du modèle SPICE d'un capteur actionneur piézoélectrique dans Design Architect.

```
.subckt piezo1 mec el_p el_n
L_L1 mec 0  7.69231e-08
R_Rc n1 el_n  1e13
R_Ro el_p n1  0.01
C_C el_p n1 7.2e-9
R_Mec mec 0  0.01
C_Cm mec 0 0.03
G_G1 0 mec POLY(1) el_p n1 0 7.8
G_G2 n1 el_p POLY(1) mec 0 0 -7.8
.ends piezo1
```

C.8 Modèle HDL-A d'un capt. action. piézoélectrique

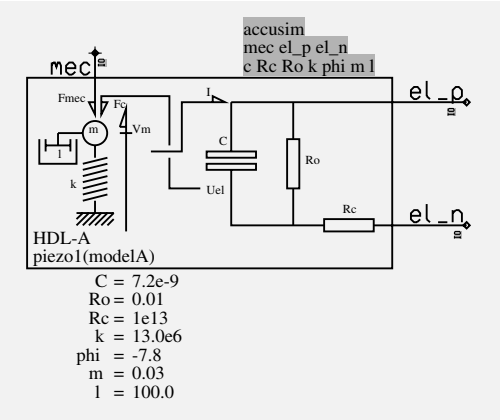


Figure B-9. Symbole du modèle HDL-A d'un capteur actionneur piézoélectrique dans Design Architect.

```
ENTITY piezo1 IS
  GENERIC (c,Rc,Ro,k,phi,m,l  : real);
  PIN     (mec                : mechanical1;
           el_p,el_n          : electrical);
END ENTITY piezo1 ;

ARCHITECTURE modelA OF piezo1 IS
  STATE v,uel,uin,Fmec,iel :analog;
BEGIN
  RELATION
  PROCEDURAL FOR dc =>
    v              := mec.tv;
    uin            := [el_p,el_n].v;
    Fmec           := l*v+phi*uel;
    iel            := (uin-uel)/Rc;
    mec.f          %= Fmec;
    [el_p,el_n].i %= iel;
  PROCEDURAL FOR ac,transient =>
    v              := mec.tv;
    uin            := [el_p,el_n].v;
    Fmec           := m*ddt(v)+k*integ(v)+l*v+phi*uel;
    iel            := (uin-uel)/Rc;
    mec.f          %= Fmec;
    [el_p,el_n].i %= iel;
  EQUATION (uel) FOR dc,ac,transient =>
    uin+Rc*phi*(v)==uel*(1.0+Rc/Ro)+Rc*c*ddt(uel);
  END RELATION;
END ARCHITECTURE modelA;
```

C.9 Modèle HDL-A du rotor ultrasonique

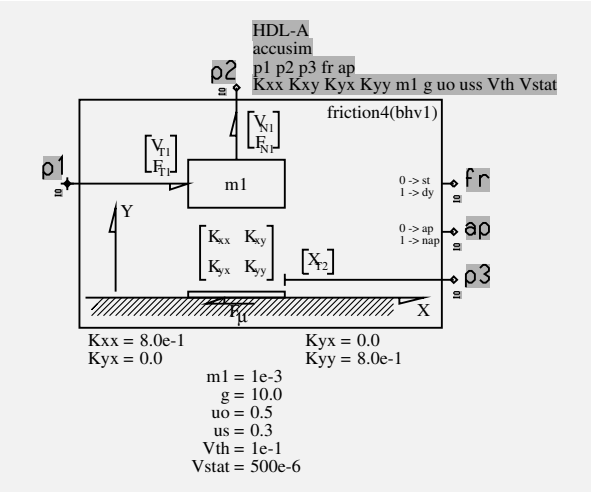


Figure B-10. Symbole du modèle HDL-A du rotor ultrasonique dans Design Architect.

```
ENTITY friction4 IS
  GENERIC (Kxx, Kxy, Kyx, Kyy   : real;
           m1, g, uo, uss, Vth : real;
           Vstat                : real);
  PIN     (p1, p2               : mechanical1;
           p3                   : mechanical2;
           fr, ap               : electrical);
END ENTITY friction4;

ARCHITECTURE bhv1 OF friction4 IS
  STATE   FT1,FN1,FT,FN,x1o,x1,y1o,y1,x2   : analog;
  STATE   dx1,dy1,ud,flag1,flag2,sgndx1    : analog;
  STATE   sgndx2,det                       : analog;
BEGIN
  RELATION
  PROCEDURAL FOR dc =>
    FT1  := p1.f;
    FN1  := p2.f;
    FN   := FN1-m1*g;
    FT   := FT1;
    det  := Kyy*Kxx-Kxy*Kyx;
    x1   := (FT*Kyy-FN*Kxy)/det;
    y1   := (FN*Kxx-FT*Kyx)/det;
    x2   := 0.0;
    x1o  := x1;
    y1o  := y1;
    flag1:= 0.0;
    IF (FN1-m1*g<=0.0)
    THEN flag2 := 0.0; -- appui
    ELSE flag2 := 1.0; -- pas d'appui
    END IF;
    p1.tv %= 0.0;
    p2.tv %= 0.0;
```

```
  p3.d %= x2;
  fr.v %= flag1;
  ap.v %= flag2;
PROCEDURAL FOR ac,transient =>
  FT1 := p1.f;
  FN1 := p2.f;
  x1  := integ(dx1)+x1o;
  y1  := integ(dy1)+y1o;
  IF (y1<=0.0)
  THEN flag2:=0.0;  -- appui
  ELSE flag2:=1.0;  -- pas d'appui
  END IF;
  IF (flag2=0.0)
  THEN -- appui
    IF (flag1=0.0)
    THEN -- Frottement statique
      FN := Kyx*(x1-x2)+Kyy*y1;
      FT := Kxx*(x1-x2)+Kxy*y1;
      IF (abs(FT)>abs(uss*FN))
      THEN
        flag1 := 1.0;
        IF dx1>=0.0
        THEN sgndx1:=1.0;
        ELSE sgndx1:=-1.0;
        END IF;
      END IF;
    ELSE -- Frottement dynamique
      ud := uo-uo*exp(-abs(dx1/Vth));
      IF (dx1>=0.0) THEN sgndx2:= 1.0;
                    ELSE sgndx2:=-1.0;
      END IF;
      x2 := (-Kxx*x1-Kxy*y1+sgndx2*ud*(kyx*x1+Kyy*y1))/
            (sgndx2*ud*Kyx-Kxx);
      FN := Kyx*(x1-x2)+Kyy*y1;
      FT :=-sgndx2*ud*FN;
      IF (sgndx1*dx1<=Vstat)
      THEN flag1:=0.0;
      END IF;
    END IF;
  ELSE -- pas d'appui
    FN := 0.0;
    FT := 0.0;
    x2 := x1;
    IF (dx1>=0.0)
    THEN sgndx1:=1.0;
    ELSE sgndx1:=-1.0;
    END IF;
    IF (sgndx1*dx1<=Vstat)
    THEN flag1:=0.0;
    ELSE flag1:=1.0;
    END IF;
  END IF;
p1.tv %= dx1;
p2.tv %= dy1;
p3.d %= x2;
```

```
  fr.v %= flag1;
  ap.v %= flag2;
  EQUATION (dx1,dy1) FOR ac,transient =>
      m1*ddt(dx1)==FT1-FT;
      m1*ddt(dy1)==FN1-FN-m1*g;
  END RELATION;
END ARCHITECTURE bhv1;
```

C.10 Modèle HDL-A et C d'un banc de tests

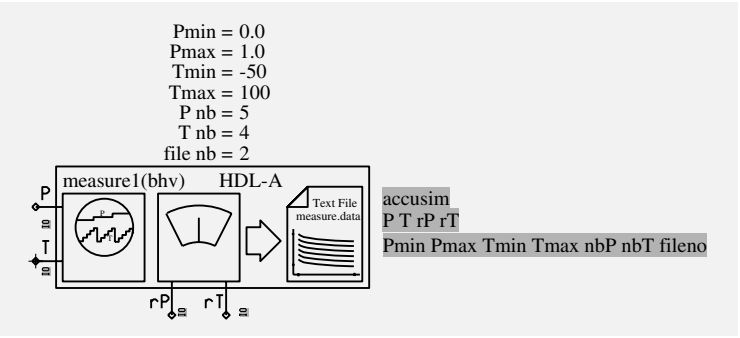


Figure B-11. Symbole du modèle HDL-A et C d'un banc de tests dans Design Architect.

C.10.1 Partie codée avec HDL-A

```
ENTITY measure1 IS
  GENERIC (Pmin,Pmax,Tmin,Tmax :real;
           nbP,nbT,fileno      :integer);
  PIN     (P                   :fluid;
           T                   :thermal;
           rP,rT               :electrical);
END ENTITY measure1;

ARCHITECTURE bhv OF measure1 IS
  VARIABLE TimeStep                          : time;
  VARIABLE count1,count2,flag1,flag2,flag3   : integer;
  VARIABLE VTm,VPm,Pm,Tm                     : real;
  VARIABLE Tstep,Pstep                       : real;
  VARIABLE Tout,Pout                         : real;
  SIGNAL   clk1                              : bit;

  PROCEDURE WRITEMEASURES3 (Coeffno:integer);
  PROCEDURE WRITEMEASURES4 (Coeffno:integer;
                            Pm,Tm,VPm,VTm:real;
                            Flag:integer);
BEGIN
  RELATION
  PROCEDURAL FOR transient =>
    T.v    %= Tout;
    P.v    %= Pout;
```

```
  END RELATION;

  PROCESS
  BEGIN
    TimeStep := 1000ms/(nbP*nbT);
    Tstep    := (Tmax-Tmin)/REAL(nbT-1);
    Pstep    := (Pmax-Pmin)/REAL(nbP-1);
    count1   := 1;
    count2   := 1;
    flag1    := 1;
    flag2    := 1;
    flag3    := 0;
    Tout     := Tmin;
    Pout     := Pmin;
    clk1     <= '0';
    WRITEMEASURES3 (fileno);
    LOOP
      WAIT FOR (TimeStep/2.0);
        IF flag1 = 0
        THEN flag1 := 1;
          IF count1 < nbT
          THEN Tout   := Tmin+REAL(count1)*Tstep;
               count1 := count1+1;
          ELSE flag2  := 0;
               Tout   := Tmin;
               count1 := 1;
               Pout   := Pmin+REAL(count2)*Pstep;
               count2 := count2+1;
          END IF;
        ELSE flag1 := 0;
          IF flag2 = 1
          THEN  flag3 := 1111;
          ELSE  flag3 := 1110;
          END IF;
          VTm   := REAL(rT.v);
          VPm   := REAL(rP.v);
          Tm    := REAL(T.v);
          Pm    := REAL(P.v);
          WRITEMEASURES4 (fileno,Pm,Tm,VPm,VTm,flag3);
        END IF;
    END LOOP;
  END PROCESS;
END ARCHITECTURE bhv;
```

C.10.2 Partie codée en C

```
#include "fci.h"
#include <math.h>
#include <stdio.h>
#include <string.h>
/*================================================*/
void  WRITEMEASURES1(char   *FileName)
{
```

```
 FILE     *fout;
 fout = fopen(FileName,"w");
 fprintf(fout, "PTVPVT\n");
 fprintf(fout, "==================================\n");
 fclose(fout);
}
/*================================================*/
void  WRITEMEASURES2(char   *FileName,
                     double P,
                     double T,
                     double VP,
                     double VT,
                     int    flag)
{
 FILE     *fout;
 fout = fopen(FileName,"a");
 if (flag==1111)      {fprintf(fout, "%+20.20lf%+20.20lf
                       %+20.20lf%+20.20lf\n",P,T,VP,VT);}
 else if (flag==1110) {fprintf(fout, "%+20.20lf%+20.20lf
                       %+20.20lfx\n",P,T,VP);}
 else if (flag==0101) {fprintf(fout, "x%+20.20lfx
                      %+20.20lf\n",T,VT);}
 fclose(fout);
}
/*================================================*/
void WRITEMEASURES3 ( fci_type  *CoeffnoType,
                      fci_value *CoeffnoValue )
{
int         Coeffno;
char        filename[255];
  Coeffno  = fci_get_integer(CoeffnoValue);
  sprintf(filename,"/data/measures%i.data",Coeffno);
  printf("//  ***** Name of the file containing the
          measures\n//  %s\n",filename);
  WRITEMEASURES1(filename);
}
/*================================================*/
void WRITEMEASURES4 ( fci_type  *CoeffnoType,
                      fci_value *CoeffnoValue,
                      fci_type  *PmType,
                      fci_value *PmValue,
                      fci_type  *TmType,
                      fci_value *TmValue,
                      fci_type  *VPmType,
                      fci_value *VPmValue,
                      fci_type  *VTmType,
                      fci_value *VTmValue,
                      fci_type  *FlagType,
                      fci_value *FlagValue   )
{
int         Coeffno,Flag;
char        filename[255];
double      Pm,Tm,VPm,VTm;
  Coeffno  = fci_get_integer(CoeffnoValue);
  Flag     = fci_get_integer(FlagValue);
```

```
  Pm        = fci_get_real(PmValue);
  Tm        = fci_get_real(TmValue);
  VPm       = fci_get_real(VPmValue);
  VTm       = fci_get_real(VTmValue);
  sprintf(filename,"/data/measures%i.data",Coeffno);
  WRITEMEASURES2(filename,Pm,Tm,VPm,VTm,Flag);
}
```

Index

N

O

P

Q

R

S