

THÈSE

présentée pour obtenir le grade de

Docteur de l'Université Louis Pasteur - Strasbourg I

Discipline : Électronique, Électrotechnique et Automatique

Spécialité : Instrumentation et Microélectronique

par

Lingfei ZHOU

**Modélisation VHDL-AMS multi-domaines de structures intelligentes,
autonomes et distribuées à base de MEMS**

soutenue publiquement le 27 novembre 2007 devant le jury :

Directeur de thèse :	Yannick HERVE	Maître de Conférences - HDR, ULP, Strasbourg
Rapporteur interne :	Daniel MATHIOT	Professeur, ULP, Strasbourg
Rapporteur externe :	Patrick GARDA	Professeur, UPMC, Paris
Rapporteur externe :	Salvador MIR	Directeur de Recherche - HDR, TIMA, Grenoble
Examineur :	Nadine PIAT-LE FORT	Professeur, ENSMM, Besançon
Examineur :	Yves-André CHAPUIS	Maître de Conférences, ULP, Strasbourg

Remerciements

Le travail présenté dans ce mémoire a été effectué respectivement au **Laboratoire d'Électronique et de Physique des Systèmes Instrumentaux de Strasbourg (Ex-LEPSI)** et à l'**Institut d'Électronique du Solide et des Systèmes (InESS)**.

Je remercie tout d'abord **Monsieur Jean-Louis RIESTER**, le directeur d'Ex-LEPSI et **Monsieur Daniel MATHIOT**, le directeur de l'InESS de m'avoir accueilli au sein de leur laboratoire. Je dois remercier particulièrement **Monsieur Daniel MATHIOT** d'avoir accepté d'être rapporteur et membre du jury.

Je remercie **Monsieur Francis BRAUN** qui était mon directeur de thèse lors de ma première étape de thèse.

Je remercie mon directeur de thèse **Monsieur Yannick HERVE**, pour m'avoir donné une vue la plus haut niveau du monde scientifique, donnant à la thèse une autre dimension. Sa culture scientifique et ses conseils pertinents me sont toujours bénéfiques.

Je tiens à exprimer tout particulièrement ma reconnaissance à **Monsieur Yves-André CHAPUIS** sans qui la thèse ne verrait jamais le jour. Les soutiens qu'il a apportés dans le déroulement et dans la rédaction de thèse sont vraiment indispensables.

Je remercie **Monsieur Patrick GARDA**, professeur à l'UPMC, **Monsieur Salvador MIR**, directeur de recherche au TIMA et **Madame Nadine PIAT-LE FORT**, professeur à l'ENSEMM d'avoir accepté respectivement d'être rapporteur et membre du jury.

Je remercie mes collègues de travail, surtout **Gyasi, Joris et Awa**, pour leur soutien et leur amitié.

Je remercie mon frère **Lingchuan**, pour son soutien permanent et efficace.

Je remercie mes parents, pour s'être occupés de moi physiquement et moralement durant ma thèse.

Table des matières

Remerciements	2
Table des matières	4
Préambule	1
1 Introduction à la conception de « Smart Structures » à base de MEMS	5
1.1 Comment programmer un nuage de poussières?	5
1.1.1 Technologie MEMS et nouveaux défis	5
1.1.2 Part du calcul (Informatique) dans le domaine des MEMS	6
1.1.3 Positionnement de recherche	7
1.2 Le monde merveilleux des MEMS	8
1.2.1 État de l’art	8
1.2.2 Applications dans le domaine des MEMS	11
1.2.3 Les MEMS distribués	14
1.2.4 Tendances et actions de recherche	16
1.3 Notions de « Smart Structure »	17
1.3.1 Définitions	17
1.3.2 Interprétation du terme « smart »	18
1.3.3 Exemples de « Smart Structure »	19
1.4 Contrôle distribué	20
1.4.1 Topologie	20
1.4.2 Notions d’agent et multi-agents	23
1.5 Simulation multi-domaines	25
1.5.1 Approche de modélisation	25
1.5.2 La modélisation par éléments finis	26
1.5.3 Modélisation analytique	26
1.5.4 Modélisation descriptive	27
1.6 Conclusion	29

2	« Smart Surface » à base de MEMS : État de l'art et prototype pneumatique	31
2.1	État de l'art	31
2.1.1	Introduction	31
2.1.2	Concepts de micromanipulation distribuée	32
2.1.3	Technologies des micromanipulateurs distribués à base de MEMS	34
2.1.4	Tendances	38
2.2	Prototype à champ de forces fluidiques	40
2.2.1	Introduction	40
2.2.2	Approche de conception	41
2.2.3	Approche de contrôle	45
2.2.4	Validation expérimentale : Caractérisation	51
2.3	Conclusion	53
3	Approche de modélisation et simulation par Prototypage Virtuel Fonctionnel	55
3.1	Introduction	55
3.2	Prototypage Virtuel Fonctionnel	56
3.2.1	Méthode de conception dite du « Cycle en V »	56
3.2.2	Le Prototypage Virtuel Fonctionnel	58
3.2.3	Approche de modélisation	60
3.3	Les langages de modélisation descriptive	62
3.3.1	Le langage VHDL-AMS	62
3.3.2	Les autres langages de modélisation descriptive	65
3.4	Outils de simulation VHDL-AMS	70
3.4.1	Environnement de simulation VHDL-AMS	70
3.4.2	Principaux outils de simulation	72
3.4.3	Simulateur SMASH (Dolphin-integration®)	73
3.4.4	Contexte de thèse	74
3.5	Conclusion	75
4	Modèle comportemental	77
4.1	Approche de description	77
4.1.1	Description générale	77
4.1.2	Description en sous-modèles	78
4.2	Éléments de mécanique des fluides	80
4.2.1	Introduction	80
4.2.2	Définitions d'un écoulement fluide	80
4.2.3	Efforts exercés par un écoulement sur un solide	82
4.2.4	Forces de « traînée » et de « portance »	84
4.3	Modèle physique de la matrice	92
4.3.1	Modèle en lévitation	92

4.3.2	Modèle de convoyage en lévitation	95
4.3.3	Modèle dynamique	98
4.4	Description VHDL-AMS	99
4.4.1	Arborescence	99
4.4.2	Structure Générale	99
4.4.3	Description du modèle « Lévitation : Cas Général »	102
4.4.4	Programme du modèle « Convoyage à 1-D en lévitation »	104
4.4.5	Programme du modèle « Convoyage 2-D en lévitation »	108
4.5	Simulation	111
4.5.1	Outils et paramètres	111
4.5.2	Cas du « convoyage 1-D en lévitation »	111
4.5.3	Convoyage 2-D en lévitation : Cas en boucle ouverte	112
4.6	Conclusion et perspective	114
5	« Modèle structurel comportemental » et « modèles composants »	117
5.1	Approche de modélisation	117
5.1.1	Préambule	117
5.1.2	Flot de modélisation	118
5.1.3	Interface	120
5.2	Modèle structurel comportemental	121
5.2.1	Étape « identification »	121
5.2.2	Étape « décomposition »	123
5.2.3	Étape « interconnexion »	125
5.2.4	Quadrillage	128
5.2.5	Description VHDL-AMS	129
5.2.6	Le corps d'architecture	129
5.3	Modèles composants	129
5.3.1	Modèle « microactionneur pneumatique »	131
5.3.2	Description VHDL-AMS	134
5.4	Résultats de simulation	134
5.4.1	Conditions de simulation	134
5.4.2	Modèle « microactionneur pneumatique »	136
5.4.3	Modèle « surface active »	137
5.4.4	Modèle « Surface Active » incluant le composant « Microactionneur »	138
5.5	Conclusion	140
6	Méthodologie de validation VHDL-AMS d'un contrôleur décentralisé VHDL	143
6.1	Objectifs et principes	143
6.1.1	Objectifs	143
6.1.2	Approche de conception du contrôleur	144

6.1.3	Besoin d'un « outil » de validation fonctionnelle	145
6.2	Méthodologie de validation	146
6.2.1	Le flot de validation mixte	146
6.3	Contrôleur décentralisé	148
6.3.1	Contrôle distribué : État du domaine	148
6.4	Validation Fonctionnelle par Simulation	152
6.4.1	Adaptation du Modèle VHDL-AMS	152
6.4.2	Niveau Algorithmique	154
6.4.3	Niveau de Transfert de Registre (RTL)	156
6.4.4	Niveau Physique (ou porte)	163
6.5	Vérification expérimentale	164
6.5.1	FPGA et plateforme	164
6.6	Conclusion	170
Conclusions et perspectives		170
Références		175
Liste des publications		182
A Codes source VHDL-AMS : modèle structurel comportemental		187
B Études de la micro-valve : La tension $V_{pull-in}$: comparaison et analyse		207
B.1	Les formules de $V_{pull-in}$ pour les trois formes	207
B.2	Les résultats de simulation	207
B.2.1	Forme 1	208
B.2.2	Forme 2	208
B.2.3	Forme 3	209
B.3	Analyse des résultats	210
B.3.1	Le bilan récapitulatif des résultats	210
B.3.2	Analyse	211

Préambule

Contexte

Un microsystème est un ensemble de dispositifs hautement intégrés, comprenant des transducteurs (capteurs et actionneurs) avec leurs interfaces électroniques de traitement/processeurs pouvant exécuter des tâches dites intelligentes en réponse à un ensemble de commandes. Avec l'avènement des technologies MEMS (*Micro Electro Mechanical Systems*), ces composants peuvent atteindre aujourd'hui des niveaux de miniaturisation submicronique. On peut également distribuer, ou mettre en réseau, un grand nombre de ces composants de très petite taille pour obtenir, par coordination active, des fonctionnalités d'ensemble différentes de celles d'origine. On peut aussi combiner, sur un même substrat, des éléments d'un microsystème incluant des parties processeurs, ouvrant ainsi des perspectives considérables dans le domaine des microsystèmes.

Le futur de ces dispositifs est aussi d'être plus distribués à travers l'environnement, couvrant ainsi des surfaces ou étant embarqués dans des objets de tous les jours pour créer des systèmes appliqués à la perception et au raisonnement. Ainsi, demain ces composants se permettront de réagir au monde physique, qui les entoure, sur une échelle encore jamais atteinte auparavant. Cependant, la complexité croissante de ces structures à base de MEMS ne va pas de paire avec une méthode de description orientée vers une seule physique. Elle implique que de tels systèmes aient des caractéristiques multi-physiques et multi-technologiques, lesquelles nécessitent des moyens de description et de modélisation relevant encore du défi technologique.

Orientations de recherche

Dans cette thèse, nous proposerons une contribution au thème de la modélisation multi-domaines pour dispositifs à base de MEMS distribués. Une étude encore faiblement développée, à notre connaissance, et pourtant éminemment importante dans la perspective de conception des composants du futur.

Nous appliquerons notre approche de modélisation à un dispositif MEMS constituant une surface active à base de microactionneurs pneumatiques en réseau pour des opérations de micromanipulation fluide « sans contact », comme le montre la figure ci-dessous.

Ce composant de micromanipulation a été conçu et fabriqué à l'Institute of Industrial Science

(IIS) de l'Université de Tokyo au sein du laboratoire du *Pr. Fujita* (Fujita Lab.) Nous bénéficions des résultats expérimentaux de ce dispositif en collaboration avec *Y.-A. Chapuis*, chercheur détaché au CNRS au sein du LIMMS/CNRS-IIS, qui a étudié et contribué à l'élaboration de ce composant.

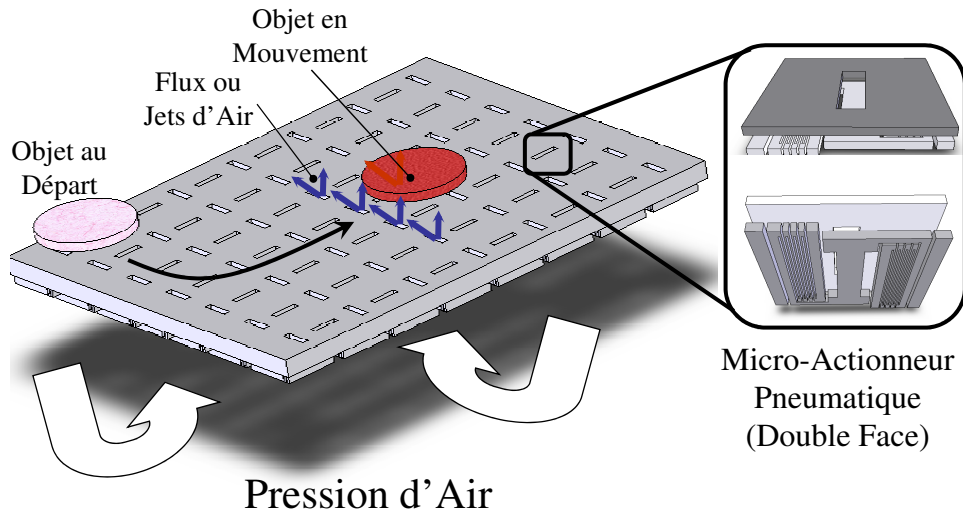


Fig. 1 – Dispositif de « micromanipulation distribuée » à base de microactionneurs pneumatiques en réseau en technologie MEMS

Ce dispositif devant, à termes, pouvoir intégrer des éléments d'autonomie et de décision pour constituer une « smart structure », une première étude de conception de contrôle a été réalisée. Il s'est agi de concevoir, à partir du langage VHDL, un modèle de contrôle décentralisé pour la commande du micromanipulateur distribué. Cette conception sera intégrée sur une cible FPGA (Field Programmable Gate Array) et validée sur la plateforme expérimentale du Fujita Lab. au Japon.

Suite à ces premiers travaux basés sur une méthodologie très faible, nous avons engagé des recherches sur les modèles de simulation multi-domaine avec pour objectif de modéliser l'ensemble « micromanipulateur et commande » pour des études de fonctionnalité du système. Face à la modélisation de tels systèmes, les processus de conception actuels ne répondent plus efficacement à ces besoins. Notre approche se basera alors sur des technologies de conception, tel que le « Prototypage Virtuel Fonctionnel ». Cette approche, qui est une nouvelle méthodologie de conception, formalisée par *Yannick Hervé*, permettant de répondre de la méthode classique de conception dite du « cycle en V ».

Les principes du « Prototypage Virtuel Fonctionnel » se basent sur le renforcement de chaque étape du précédent « cycle en V » par des modèles fonctionnels décrits à un niveau d'abstraction bien choisi, comme le montre la figure 2.

La mise en place de la méthodologie de « Prototypage Virtuel Fonctionnel » nécessite des outils de supports performants. Pourtant, peu de langages ou de méthodes de descriptions, qui se disent pourtant multi-domaines, peuvent véritablement décrire et simuler la complexité

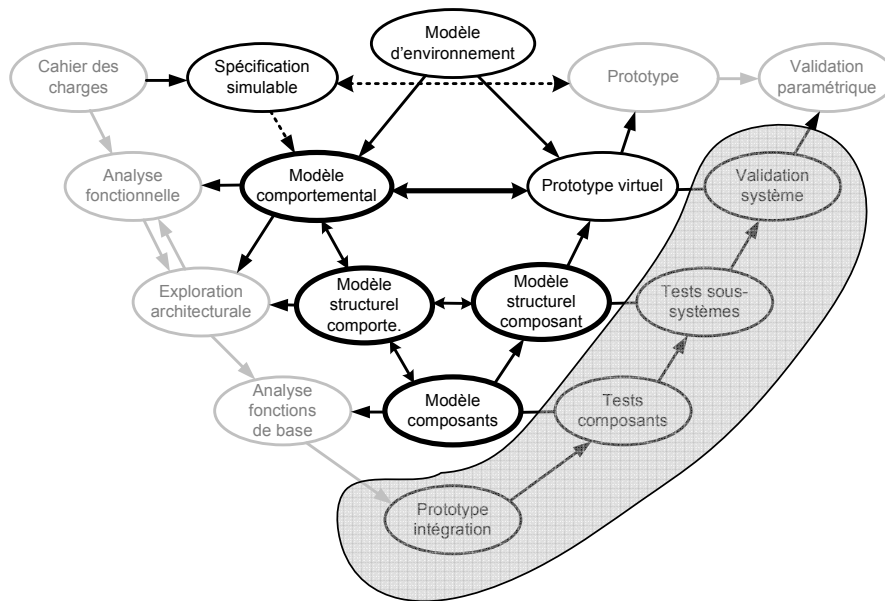


Fig. 2 – *Cycle de prototypage virtuel fonctionnel*

de telles structures impliquant des interactions entre et dans les MEMS mais également avec leur environnement physique ou fonctionnel : ils ne sont tout simplement pas assez puissants sémiologiquement. Après une étude sur les langages et les outils de modélisation, le langage VHDL-AMS a été choisi.

Cette thèse est composée de six chapitres pour couvrir l'ensemble des aspects abordés :

Chapitres de thèse

Le **chapitre I** commence par présenter la problématique de recherche dans le domaine de la modélisation et du contrôle de « smart structures » à base de MEMS. On rappellera aussi l'intérêt grandissant des technologies MEMS dans le domaine industriel et des limites de conception actuelles dans le domaine des structures distribuées multi-technologiques.

Le **chapitre II** sera consacré à l'état de l'art des dispositifs type « smart structure », à base de MEMS, appliqués en « micromanipulation distribuée ». Plus spécifiquement, nous étudierons un prototype, inspiré des travaux de l'Université de Tokyo (Japon), lequel se présente comme un réseau de micro-actionneurs pneumatiques pour la génération de forces fluidiques distribuées. Ce prototype et son environnement seront les éléments de base du modèle que nous aurons à décrire à l'issue de cette thèse.

Le **chapitre III** introduira l'approche de modélisation VHDL-AMS que nous allons adopter dans cette thèse, laquelle s'inspire du cycle de « Prototypage Virtuel Fonctionnel », que nous décrivons ici. Les outils et moyens de programmation et de simulation mis en oeuvre seront aussi traités dans ce chapitre.

Le **chapitre IV** sera dédié à la première phase de modélisation du dispositif de « micro-manipulation fluide distribué ». Nous décrirons ainsi un « modèle comportemental » au plus haut niveau d'abstraction. Des premiers résultats de simulation seront alors comparés avec ceux obtenus expérimentalement.

Le **chapitre V** sera consacré à la seconde phase de modélisation. Nous décrirons ainsi le « modèle structurel comportemental » et les « modèles composants » de notre dispositif à base de MEMS. Dans une première étape, nous élaborerons les modes de décomposition du « modèle structurel comportemental » dans lequel nous intégrerons le « modèle composant » du « microactionneur pneumatique ». L'ensemble sera testé en simulation validant avec succès l'approche de conception.

Le **chapitre VI** correspondra à la phase de test de la méthodologie que nous élaborerons à partir des modèles VHDL-AMS de validation. Nous appliquerons cette méthodologie dans le cas d'un modèle de contrôleur décentralisé que nous validerons dans le flot de conception du « niveau algorithmique » jusqu'au « niveau physique ». La cible d'intégration choisie sera un composant FPGA. Notre commande sera implantée sur la plateforme expérimentale du micromanipulateur distribué installée au sein du Fujita Lab. au Japon.

Chapitre 1

Introduction à la conception de « Smart Structures » à base de MEMS

1.1 Comment programmer un nuage de poussières ?

1.1.1 Technologie MEMS et nouveaux défis

En 1997, *Berlin et Gabriel*, tous deux chercheurs respectivement au *Xerox Palo Alto Research Center* et à la *Electronics Technology Office (U.S. Def. Adv. Res. Proj. Ag.)*, publiaient dans la revue « *IEEE Computational Science and Engineering Journal* » un article où ils exprimaient leur interrogation face au futur défi « *informatique* » (en anglais « *computational* ») de structures complexes et intelligentes à base de MEMS distribués (ou en réseau) : « *Distributed MEMS : New Challenges for Computation* » [1].

La question qu'ils envoyaient à la communauté scientifique de l'époque était liée à l'apparition des MEMS (Microelectromechanical systems) et de leur intégration dans les systèmes et structures du futur. En effet, la technologie à base de MEMS était déjà amenée à remplacer les composants à base de microélectronique, pourtant en plein essor à cette époque. Si les problématiques de « *calcul* » et de leur intégration dans le composant étaient parfaitement maîtrisées dans le cas de ces systèmes microélectroniques, c'était encore loin d'être le cas pour les MEMS qui associaient des fonctions à la fois de *capteur*, de *calcul* et d'*actionneur*.

Pour illustrer leurs propos, *Berlin et Gabriel* posaient cette question très imagée : « *How do you program a cloud of dust ?* » (*Comment programmer un nuage de poussières ?*), prenant l'exemple d'une aile d'avion (*aircraft wing*) couverte de composants MEMS communicants en réseau, comme le montre la figure 1.1. Chacun de ces composants pouvait être considéré comme un MEMS intégrant les trois éléments de base suivants :

- Capteur (*S* : *Sensor*),
- Calcul (*informatique*) (*C* : *Computation*)
- Actionneur (*A* : *Actuator*),

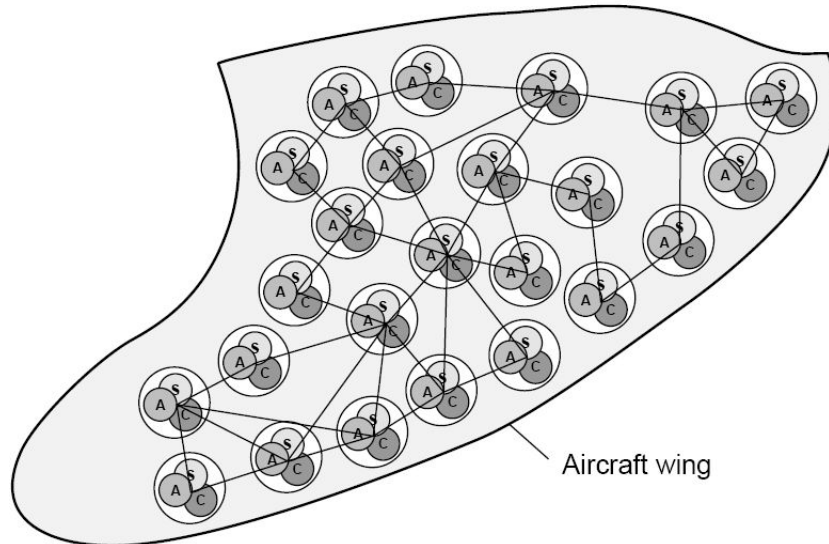


Fig. 1.1 – Structure d’aile d’avion (*aircraft wing*) à base de MEMS distribués (image extraite de l’article [1])

Ce que ces chercheurs exprimaient dans cette description était l’implication à termes de modules « capteur-calcul-actionneur » dans des structures « intelligentes », « autonomes » et « distribuées » à base de MEMS pour capter le monde physique et agir sur lui. Dans leur exemple, on voit que de minuscules volants embarqués dans la surface de l’aile d’un avion pourraient réduire la poussée nécessaire de l’appareil en captant les « *vortices* » et en agissant sur eux.

Berlin et Gabriel souhaitaient ainsi poser les bases des futurs défis technologiques qu’attendaient leurs collègues chercheurs de par le monde. Ils montraient déjà que ces défis ne seraient relevés que si l’on atteignait la maîtrise des technologies à base de MEMS du point de vue « *informatique* ».

1.1.2 Part du calcul (*Informatique*) dans le domaine des MEMS

Le raisonnement de *Berlin et Gabriel* reposait sur plusieurs postulats. Pour eux, les MEMS lançaient des défis d’ordre « *informatique* » puisque les MEMS étaient une technologie dans laquelle des multitudes de minuscules dispositifs interagissant pouvaient ajouter un comportement « *informatique* » aux matériaux et à l’environnement en mode embarqué, massivement réparti ou distribué. En englobant *perception*, *actionnement*, *contrôle* et *communication*, les dispositifs MEMS peuvent être distribués à travers l’environnement, couverts en surface ou embarqués dans des objets de tous les jours pour créer des systèmes distribués appliqués à la perception, au raisonnement, et réagissant aux événements du monde physique sur une échelle jamais atteinte

auparavant.

Les applications des MEMS distribués devaient aller bien au-delà des limites d'échelle des paradigmes (déclinaisons) informatiques de l'époque, jetant de sérieux défis et de nouvelles opportunités pour la technologie de l'information. Ainsi, les MEMS devaient s'avancer et diriger le « *calcul* » dans les 4 domaines clés suivants :

- Le « *contrôle distribué* » ; soit le *contrôle de grand nombre de MEMS capteurs et actionneurs distribués ou répartis*,
- « *L'intelligence distribuée* » ; soit *l'intelligence distribuée rehaussant l'intelligence générale et la capacité de dispositifs et de la matière*,
- Les « *dispositifs MEMS comme éléments de calcul* »,
- « *Simulation multi-domaine* » ; soit *la simulation, l'analyse et la conception de domaines d'énergie multiple*.

De ces quatre domaines clés, *Berlin et Gabriel* n'envisageaient de répondre que sur le premier « *contrôle distribué* ». C'est d'ailleurs dans ce domaine du contrôle de MEMS capteurs et actionneurs distribués ou répartis (de milliers à des millions d'éléments) que leurs recherches ont été les plus importantes, comme pour de nombreux chercheurs de l'époque [2–4].

Le second et le troisième domaine, soit « *intelligence distribuée* » et les « *dispositifs MEMS comme éléments de calcul* », font encore aujourd'hui l'objet d'études avancées dans le domaine mathématique et algorithmique. Cependant, ces recherches restent encore peu appliquées du fait de la difficulté à modéliser et implanter de tels systèmes [5].

C'est dans le dernier domaine, évoqué par *Berlin et Gabriel* ; soit la « *simulation multi-domaines* », que de nombreux progrès ont été accomplis lors des dernières années grâce à l'avènement des outils de simulation très performant et de langages de description du matériel (Hardware Description Language) comme Verilog-AMS et VHDL-AMS [6]. Pourtant, s'il paraît possible aujourd'hui d'envisager de tels développements, il n'existe encore pas ou peu, à notre connaissance, de réalisations permettant de modéliser des niveaux de complexités telles que les structures envisagées par *Berlin et Gabriel*.

1.1.3 Positionnement de recherche

L'idée de cette thèse a pris naissance en prenant en compte cet état de fait, et la volonté de vouloir contribuer à ces recherches en développant des modèles de *simulation multi-domaine* pour la conception et la validation fonctionnelle de *structures dites intelligentes, autonomes et distribuées*, que nous nommerons « smart structure » dans la suite de ce mémoire.

Cependant, et avant de présenter nos développements de recherche, nous introduirons dans ce chapitre les notions de MEMS et de « *smart structure* », ainsi que les orientations envisagées dans les domaines du « *contrôle distribué* » et de la « *simulation multi-domaine* ».

1.2 Le monde merveilleux des MEMS

1.2.1 État de l'art

1.2.1.1 Échelle d'application

Lorsqu'on évoque les nouvelles technologies, on pense généralement aux technologies de l'information, dont Internet semble être le porte flambeau. Mais les nouvelles technologies ne concernent pas seulement l'information ; elles recouvrent un ensemble de domaines, dont certains sont appelés à occuper une place importante dans la révolution technologique que nous connaissons [7]. C'est le cas des MEMS. Les MEMS signifient « Micro Electro Mechanical Systems ». Ce sont des systèmes électromécaniques dont la taille est comprise entre un et 300 microns. La figure 1.2 donne la comparaison des ordres de grandeur rencontrés avec les MEMS.

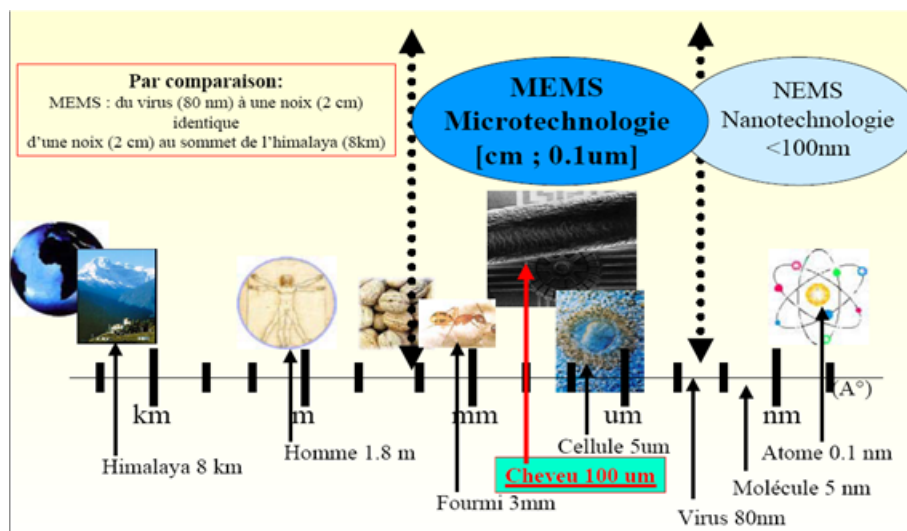


Fig. 1.2 – Comparaison des ordres de grandeur rencontrés avec les MEMS

1.2.1.2 Structure générale

Comme l'illustre la figure 1.3, si l'on devait représenter la structure générale d'un MEMS, on pourrait dire que c'est un composant composé des quatre composants de base suivants [8] :

- *Microélectroniques (MicroElectronics)*
- *Microcapteurs (MicroSensors)*
- *Microactionneurs (MicroActuators)*
- *Microstructures (MicroStructures)*

Les éléments *microélectroniques* d'un MEMS sont très similaires aux chips électroniques comme nous connaissons aujourd'hui. L'élément *microélectronique* agit comme le « cerveau » du système. Il reçoit des données, les traite, et prend des décisions. Les données reçues proviennent des éléments *microcapteurs* du MEMS.

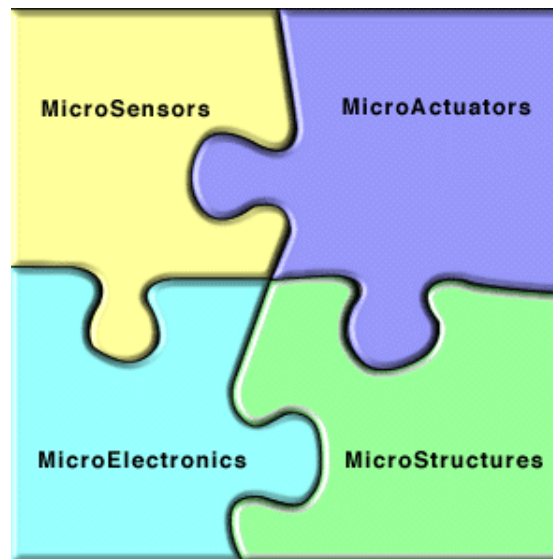


Fig. 1.3 – Structure générale d'un MEMS (image extraite de l'article [8])

Les *microcapteurs* agissent comme des bras, des yeux, un nez, etc. Ils rassemblent constamment les données venant de l'environnement ambiant et transmettent cette information aux *parties microélectroniques* pour leur traitement. Les capteurs peuvent surveiller les interprétations mécaniques, thermiques, chimiques, optiques et magnétiques à partir de l'environnement ambiant.

Un *microactionneur* agit comme un interrupteur ou un déclencheur pour activer un dispositif externe. Comme les éléments *microélectroniques* traitent les données reçues des *microcapteurs*, il prend des décisions sur « que faire ? », lesquelles sont basées sur ces informations. Parfois, la décision va impliquer l'activation d'un dispositif extérieur. Si cette décision est portée, les éléments *microélectroniques* vont dire aux *microactionneurs* d'activer le dispositif.

Grâce à la progression des technologies pour la microfabrication, des structures extrêmement petites peuvent être construites à la surface d'un chip. Ces minuscules structures sont appelées *microstructures* et sont en fait construites directement à partir du silicium des MEMS. Parmi d'autres choses, ces *microstructures* peuvent être utilisées par exemple comme valves pour contrôler le flot d'une substance ou comme de très petits filtres.

1.2.1.3 Mode de fabrication

Ces systèmes s'élaborent à partir de techniques de lithographie, dépôt et gravure, bien maîtrisées aujourd'hui, grâce au savoir faire acquis en microélectronique. Depuis dix ans, on sait fabriquer toutes sortes de MEMS de base : poutres, ressorts, conduites, leviers, roues, engrenages, etc. La figure 1.4 montre un exemple de processus de fabrication par micro-usinage en surface d'un dispositif MEMS [9].

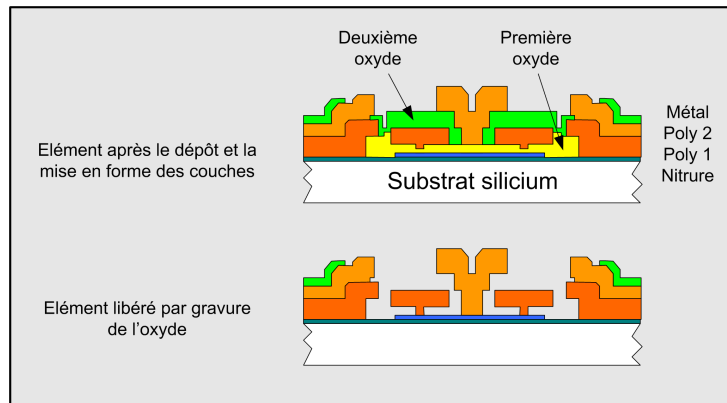


Fig. 1.4 – Exemple d'un processus de micro-usinage en surface pour la fabrication d'un dispositif MEMS. Ce processus se base sur l'emploi de couches sacrificielles (image extraite de l'article [8])

1.2.1.4 Marché des MEMS

Dans une période un peu plus récente, on a assemblé intelligemment ces MEMS pour fabriquer des microsystèmes complexes : moteurs, résonateurs, actionneurs, pompes, etc. Tous ces objets contribuent à reconstituer, à petite échelle, le monde des machines “macroscopiques” que nous connaissons. Il se fait qu'ils ont engendré une activité économique considérable : le volume se chiffre en dizaines de milliards d'euro ou dollars (\$) par an et ce chiffre est appelée à croître considérablement dans l'avenir.

Ces données ont été rapportées par le « Global Forecast of the MEMS Market, AAGRs for 2004, 2005 and 2010 in \$ Billions (*AAGR : average annual growth rate) » [10], d'où est extrait le graphique de la figure 1.5.

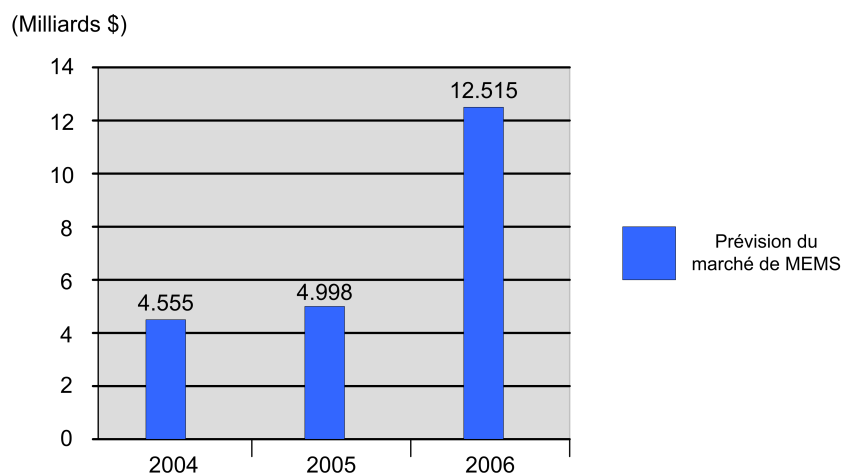


Fig. 1.5 – Marché des MEMS en milliard de dollars au cours des dernières années (adaptation du « Global Forecast of the MEMS Market », AAGRs for 2004, 2005 and 2010 in \$ Billions [10])

1.2.2 Applications dans le domaine des MEMS

Parmi les premiers microsystèmes en technologie MEMS à grand succès économique, nous citerons les MEMS pour airbag. Dans un MEMS pour *airbag*, l'unité de détection comprend deux micro-peignes imbriqués l'un dans l'autre, l'un fixe l'autre mobile. Lors d'un choc, une variation brutale de capacité apparaît. L'élément de détection, l'électronique de traitement et de déclenchement du coussin, sont intégrés sur une puce de quelques millimètres carrés, produite industriellement à très faible coût. Miniaturiser est avantageux pour la détection, mais, sur le plan économique, c'est la possibilité d'intégrer et de produire à bas coût qui explique le grand succès des MEMS pour airbag [11].

De nombreuses autres applications à succès ont vu le jour depuis, comme le célèbre vidéo projecteur de Texas Instruments, lequel se base sur une technologie MOEMS (Micro Optical Electro Mechanical Systems) pour la fabrication de DMD (Digital Mirror Devices) pour l'affichage [12]. La figure 1.6 donne les détails de fabrication et d'application de ce dispositif.

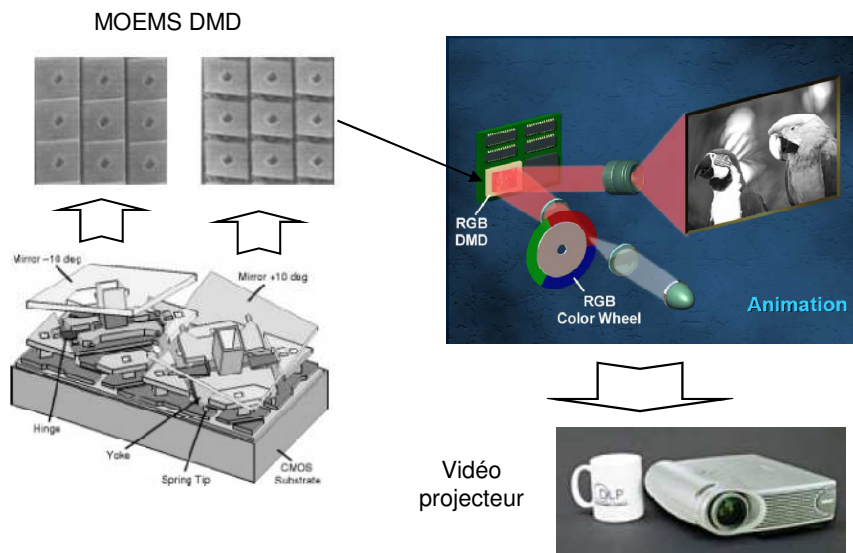


Fig. 1.6 – Le système DMD : Digital Mirror Devices ou MOEMS (images extraites de l'article [12])

Dans certains cas, la miniaturisation est non seulement avantageuse mais cruciale du point de vue de la physique. La miniaturisation permet par exemple, dans le cas des imprimantes, de délivrer de manière très précise des gouttes d'encre mono-disperses de quelques dizaines de picolitres [13]. Dans les jets ordinaires, ce type de contrôle est hors de portée du fait de la turbulence.

Demain, ce seront les domaines des télécommunications optiques, des Radio-Fréquences (RF) et des biocapteurs qui seront investis par les MEMS, engendrant pour chacun d'entre eux des petites révolutions. La figure 1.7 nous montre des exemples de réalisation de RF MEMS [14].

Nous nous arrêterons aussi sur les biocapteurs intégrés, qui méritent d'intéresser un nombre

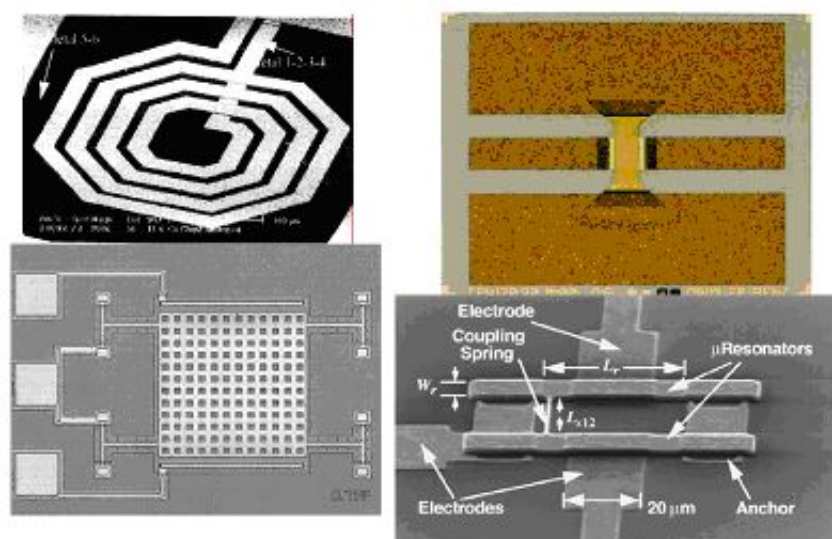


Fig. 1.7 – Exemples de réalisation de RF MEMS (images extraites de l'article [14])

de physiciens de plus en plus grand. Les biocapteurs ont pour fonction de détecter des molécules biologiques, et de les caractériser [15]. Par exemple, il s'agit d'identifier une séquence génétique sur un fragment d'ADN, permettant de révéler la présence d'un virus, ou de mettre en évidence une filiation.

Dans ce domaine particulier des puces à ADN, l'entreprise leader est californienne, mais la France développe des actions très intéressantes dont l'ampleur semble à la hauteur des enjeux. Dans ces systèmes, le fragment d'ADN doit être purifié et amplifié avant d'être placé sur la puce [16]. Il s'agit d'un travail de préparation important et il est clair qu'il serait intéressant d'intégrer toutes les opérations sur une même puce, afin de pouvoir analyser directement un échantillon brut, comme une goutte de sang ou un morceau de gruyère. Cela implique de miniaturiser des systèmes tels que cytomètres, séparateurs, bioréacteurs, et ensuite, de les associer. On définit ainsi le domaine des microsystèmes d'analyse intégrés, désigné par TAS en anglais (Micro Total Analysis Systèmes), et SAI en français (Micro Systèmes d'Analyse Intégrés) [17]. Ce domaine est en pleine effervescence aujourd'hui ; on imagine toutes sortes de SAI, analysant la qualité de l'eau, de l'air, d'aliments, reconstituant la multiplicité de fonctions couvertes par le nez, la langue ou travaillant in vivo contre la maladie d'Alzheimer ou de Parkinson [18].

La figure 1.8 montre le détail de principes et la réalisation d'un produit industrialisé d'une puce à ADN [16].

Les plus fameuses réussites de produits industriels en technologies MEMS ne sauraient nous faire oublier l'ensemble impressionnant des autres applications qui découlent des mêmes principes. Le tableau 1.1 nous donne un aperçu sélectif de ces applications dans les cinq domaines les plus en vue actuellement : La défense, le médical, l'électronique, les télécommunications et l'automobile.

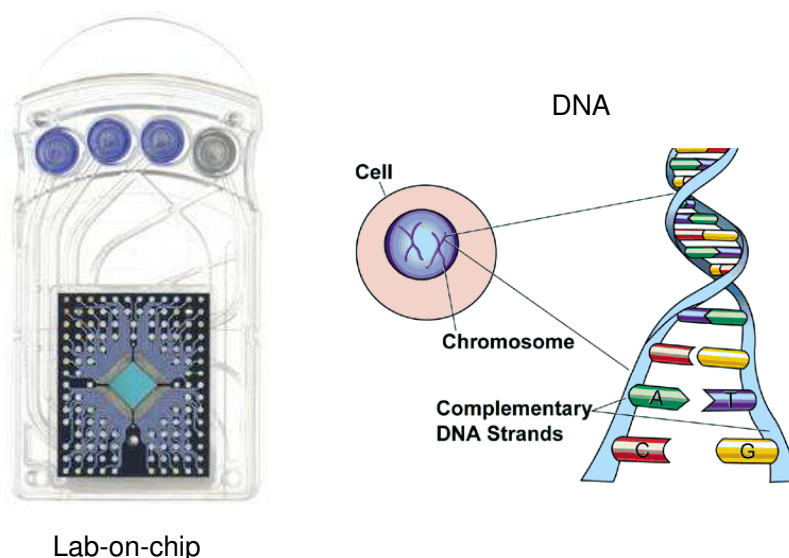


Fig. 1.8 – *Détail de principes et réalisation sous forme de produit d'une puce à ADN. (images extraites de l'article [16])*

Défense	Médical	Électronique	Communication	Automobile
Munitions Guidances	Capteurs de pression sanguine	Têtes de lecture de disque dur	Commutateurs photoniques et optiques, et raccords de réseaux large bandes	Capteurs de navigation interne
Surveillance	Systèmes de délivrance de médicaments & de stimulation de muscle	Tête d'imprimante par jets d'encre	Relais, commutateur et filtres RF	Capteur de compression pour système d'air conditionné
Systèmes d'armement	Capteurs de pression implantés	Projecteur pour écran de TV	Afficheurs de projection portable pour dispositifs de communication et d'instrumentation	Capteur de force de freinage et de suspension par contrôle par accéléromètre
Capteurs embarqués	Prothèse	Capteurs de tremblement de terre	Voltage controlled oscillators (VCOs)	Niveau de combustible et capteurs de pression « vapeur »
Stockage des données	Instruments analytique miniature	Capteurs de pression pour l'avionique	Fendeurs et coupeurs	Capteurs « airbag »
Contrôle d'avion	Stimulateur cardiaque	Systèmes de stockage massif des données	Lasers accordables	Pneus « intelligents »

Tab. 1.1 – *Applications des MEMS (exemples représentatifs, non exhaustifs)*

Comme on le voit, les développements économiques attendus dans le domaine des MEMS sont considérables. Et pourtant, les microsystèmes d'analyse intégrés mettent en scène de nombreuses situations où la physique est encore loin d'avoir établie des bases définitives : dynamique et configurations d'une molécule unique, phénomènes de mouillage et d'adhésion à l'échelle nanoscopique, interaction polymère/parois, écoulements mésoscopiques, micromélange, microfluidique, etc. Il existe donc encore de nombreuses voies d'application à explorer.

Les graphes de la figure 1.9 montre les futures tendances sur le marché des applications pour les MEMS envisagées aujourd'hui et demain [19].

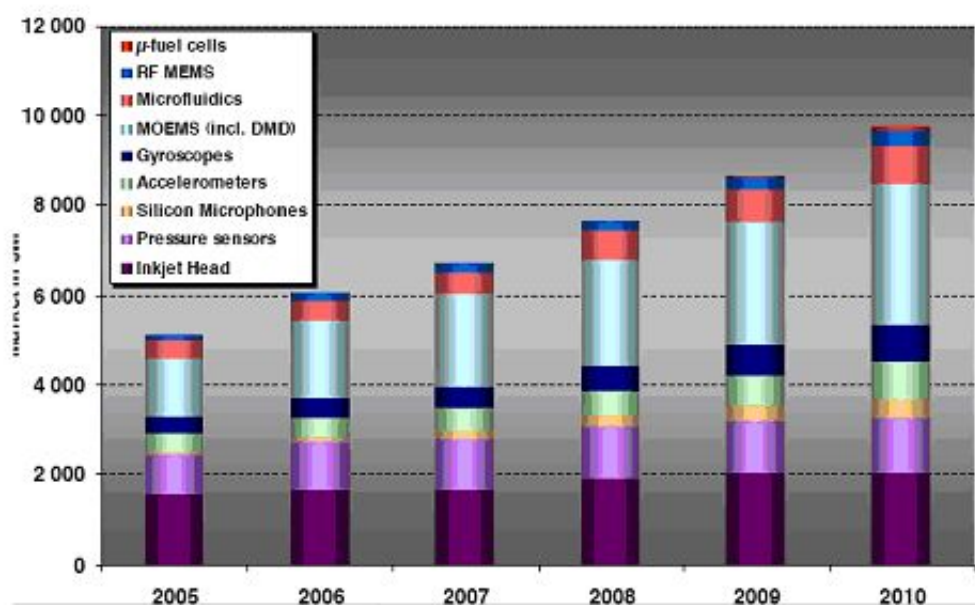


Fig. 1.9 – *Marchés et applications des MEMS (graphe extrait d'une synthèse de Yole Développement [19])*

1.2.3 Les MEMS distribués

Parmi les futures applications à base de MEMS, les systèmes ou structures dites « *distribués* » sont certainement les plus prometteuses. En effet, sachant que les technologies MEMS sont connues pour leur capacité à produire des millions de composants microscopiques au cours d'un seul et même lot de fabrication, il est aussi possible de réaliser des réseaux de plusieurs milliers et même millions de composants. Nous parlerons de « *MEMS distribués* » pour évoquer ce type de systèmes ou structures.

Les dimensions impliquées par de tels dispositifs privilégient naturellement les applications aux échelles millimétrique et/ou micrométrique et concernent, notamment, la manipulation et l'hybridation parallèle et massive des composants sur silicium dans l'industrie électronique. Mais les recherches consacrées aux réseaux de microactionneurs pourraient également avoir des réper-

cussions à l'échelle macroscopique [20].

De plus, du fait de la capacité des technologies MEMS à intégrer des éléments électroniques et mécaniques dans des structures monolithiques, on assiste aujourd'hui à l'avènement d'applications de très haute technologie dans le domaine du spatial et l'avionique, comme les réseaux de capteurs et d'actionneurs distribués sur des structures d'avion [21]. En effet, en profitant des très petites dimensions des MEMS, il est possible d'intégrer à un fuselage un réseau de capteurs qui fournissent une mesure en différents points de la pression et du flux d'air mais également des contraintes dans les matériaux ou de la température.

La figure 1.10 montre un exemple de réseaux de capteurs et d'actionneurs distribués sur la structure d'un avion de chasse (application militaire) [22].

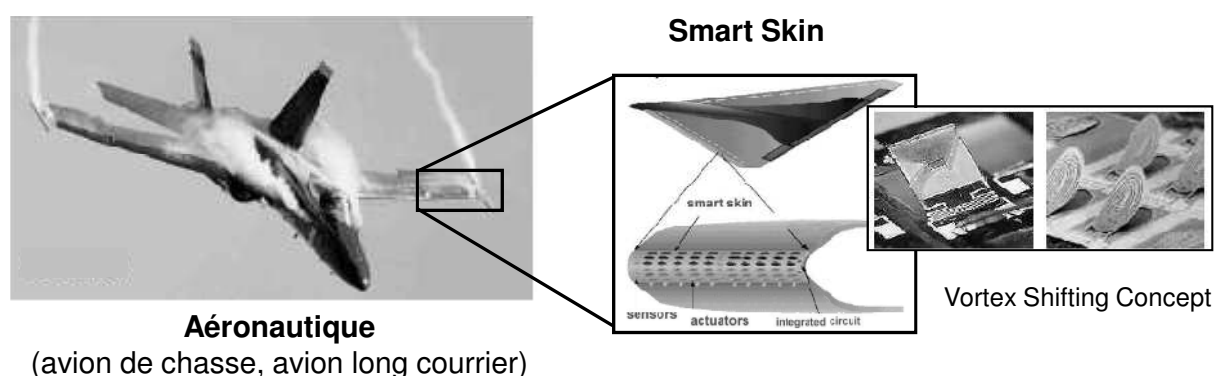


Fig. 1.10 – Exemple de réseaux de capteurs et d'actionneurs distribués sur la structure d'un avion de chasse militaire (images extraites de l'article [22])

Il existe encore beaucoup d'autres applications où les MEMS en réseaux ou *MEMS distribués* peuvent être encore appliqués, d'autant plus que les technologies MEMS permettent aujourd'hui de les rendre encore plus distribuées, autonomes et intelligentes. Une application très en vue est l'utilisation de MEMS pour des structures en réseaux de microcapteurs sans fils qui permettent l'étude du comportement des structures, avec des applications par exemple en génie civil [23], en domotique et en biomédical [24].

La figure 1.11 montre le cas d'une application en bionique par la mise en réseau de capteurs et actionneurs, communiquant entre eux sans fils [25].

D'une manière générale, ce besoin dans le domaine des MEMS distribués répond aux objectifs à moyen et long terme de nombreuses applications de haute technologie, comme dans le biomédical ou le militaire spatiale. Dans le domaine spatial militaire, par exemple, la réduction des coûts et délais, sans réduire les performances pour permettre les applications commerciales, est devenue un impératif [26]. Cela demandera une ou des percées technologiques dans plusieurs domaines comme ceux :

- Des matériaux structuraux nouveaux (buckytubes), etc.,
- De l'utilisation de produits produits en masse,
- Des systèmes de contrôle distribué plutôt que centralisé,

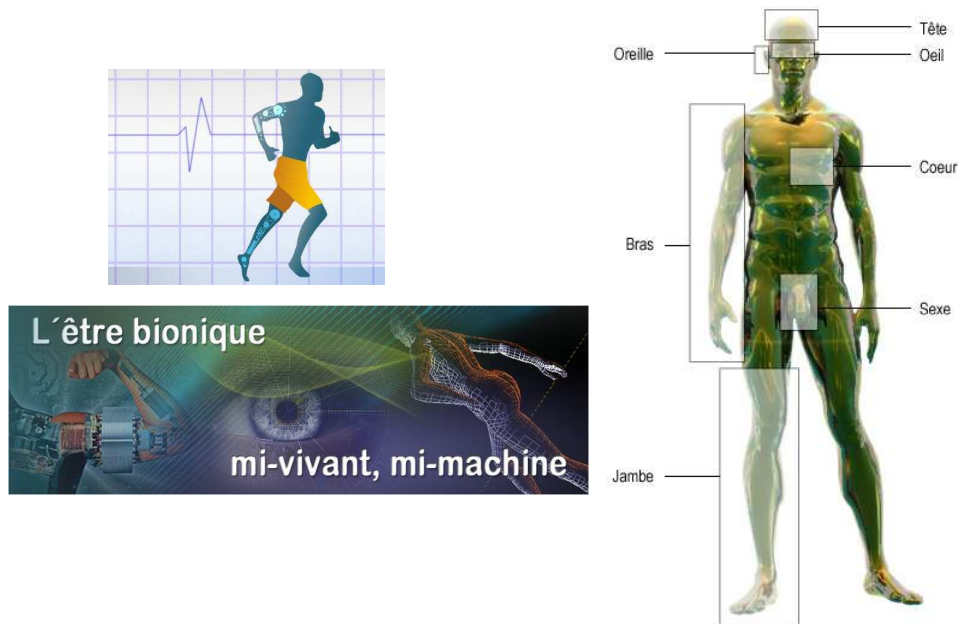


Fig. 1.11 – Application en bionique par la mise en réseau de capteurs et actionneurs (image extraite de la source [25])

- Des nouveaux concepts de satellites (utilisés en essaim),
- Des composants plus légers, plus petits, plus intelligents, consommant moins, ... (ASIM Application Specific Integrated Micro-instruments)
- Des moyens de simulation de tels systèmes ou structures distribuées.

1.2.4 Tendances et actions de recherche

Plusieurs questions se posent face à cette explosion des applications et cette fulgurante avancée des technologies MEMS dont l'avènement des dispositifs « *intelligents* », « *autonomes* » et « *distribués* » semble être l'aboutissement. La plus importante des questions est peut être la suivante : « Sommes-nous suffisamment matures technologiquement parlant pour intégrer ou même modéliser de tels éléments ? »

La réponse à cette question dépend bien évidemment de l'évolution des futures technologies de conception et de fabrication de tels dispositifs. Cependant, cette maturité technologique semble se rapprocher à grand pas et nous devons déjà être prêts pour l'anticiper.

Abordons à présent les notions de « smart structure » qui préfigurent cette thèse. En effet, comme on le verra, la définition des termes employés dans ce domaine peut souvent prêter à confusion ce qui nécessite, de notre part, une courte introduction.

1.3 Notions de « Smart Structure »

Les chercheurs de par le monde ont toujours imaginé des technologies pour implanter « de l'intelligence » dans un système ou une structure. Définie à l'origine pour des systèmes à taille conséquente, ces technologies sont en train de gagner en miniaturisation avec l'avènement des MEMS, ce qui permettra demain de répondre de manière optimale à des problèmes extrêmement complexes. On voit ainsi apparaître de plus en plus de dispositifs basés sur ces technologies et souvent intitulés sans rigueur : « systèmes intelligents », « smart structure », etc. Pourtant, les notions de « système » ou « structure » dite « intelligente » ou « smart » ont toutes une fonction et une application bien distinguée. Il est donc essentiel de bien utiliser ces termes et leur vraie définition.

1.3.1 Définitions

Pour définir les termes généraux de « *système* » et de « *structure* », nous nous référerons tout d'abord au dictionnaire scientifique [27] :

- **Système** : un groupe de dispositifs ou d'objets artificiels ou une organisation formant un réseau, en particulier pour une distribution de quelque chose ou servant une raison commune.
- **Structure** : L'agrégat d'éléments d'une entité dans leur relation avec les autres.

De même, quelques clarifications, en rapport avec les notions comme « intelligence » et « smart », sont nécessaires. Ces termes sont souvent mal interprétés et utilisés à contre-sens. De nouveau, nous nous sommes penchés sur le dictionnaire scientifique pour en extraire la signification exacte [27] :

- **Intelligent** : Ayant ou montrant un haut ou satisfaisant degré d'intelligence et capacité mentale. Avec pour « *intelligence* » la définition d'habilité à apprendre ou comprendre ou de s'adapter à de nouvelles situations.
- **Smart** : La définition anglaise du terme « *smart* » donnée dans le *Merriam websters dictionary* [27] : Fait en une partie *smart* (*Making one smart*); éveillé (*mentally alert*); brillant (*bright*), instruit (*knowledgeable*).

Ces définitions restent peut être trop « littéraires » pour distinguer les différences essentielles existantes entre elles. Il existe une différence fondamentale entre les notions « intelligent » et « smart » qui n'apparaît pas toujours hors du contexte technologique. Détaillons donc ces termes lors de leur utilisation liée au système ou à la structure.

1.3.1.1 Interprétation du terme « intelligent »

Les définitions de « *intelligence* » sont nombreuses, controversées et, en fait, il est difficile de définir précisément des domaines si vastes. Une définition, presque subconsciente et probablement due à la naturelle propension de l'homme à se considérer supérieur, pourrait être que l'intelligence

est tout ce que l'homme fait mieux que la machine. Cette définition est bien sûr évolutive au cours du temps, et même fuyante. En effet, si au début du siècle, par exemple, la capacité à calculer vite et bien, était considérée comme une preuve d'intelligence, l'avènement des ordinateurs et des calculatrices a semble-t-il rangé ce type d'exploit au rang de curiosité. Les exemples sont nombreux de progrès réalisés par l'informatique, qui réduisent ainsi le domaine de l'intelligence (car en fait, pour beaucoup, intelligence implique intelligence humaine, excluant l'identification de soi à une machine, aussi « *intelligente* » soit-elle).

On évoque la définition du terme « *intelligent* » faite, dans le cadre d'un « *capteur intelligent* » [28] par *Breckenbridge et Husson*. Ces derniers désignent ce type de composant comme étant capable de répondre automatiquement à un changement d'environnement en se basant sur un algorithme fonctionnel qui leur serait implanté. Cependant, cette *notion* d'intelligence reste assez subjective et dépend en partie de la conjoncture technologique. En effet, prenons le cas d'un *capteur intelligent*, il est de plus en plus naturel d'embarquer dans ce composant des algorithmes complexes impliquant des techniques neuromimétiques ou des systèmes experts. Demain, ces mêmes capteurs seront peut être dépassés par des composants plus intelligents et perdront donc leur terminologie de « *capteur intelligent* ». On peut évidemment imaginer une tendance similaire dans le cas de « *microsystèmes intelligents* ».

1.3.2 Interprétation du terme « smart »

Dans la littérature, on remarquera qu'une fois la notion d'intelligence définie, il s'est agi de présenter les moyens d'implantation de cette intelligence avec le microsystème. On abordera donc la notion de cible d'implantation, laquelle existe sous la forme de cible « *matérielle* » (ex : circuit SoC) ou de cible « *logicielle* » (ex : microprocesseur programmé). Pour développer notre argumentation, nous allons tout d'abord nous baser sur un composant capteur intégré dans une puce de silicium, plus communément appelé en anglais « *smart sensor* ». Ce terme a été inventé au début des années 80 par les ingénieurs électroniciens et a fini par être employé pour exprimer généralement l'intégration d'un capteur dans un même substrat avec sa circuiterie en technologie microélectronique [29].

Le terme « *smart* » est donc essentiellement utilisé dans le sens « **intégration** ». Par exemple, on parlera d'un « *système intelligent intégré* » pour exprimer un « *smart système* ». On voit donc comment séparer les notions des termes « intelligent » et « smart ».

La figure 1.12(a) montre schématiquement le concept de base d'un « *smart sensor* » où le capteur silicium intègre en partie (pré-processeur) et/ou entièrement une unité de processeur (processeur). Les figures 1.12(b) et 1.12(c) reprennent respectivement ce concept dans le cas d'un « *smart actuator* » et d'un « *smart MEMS* ».

Dans la suite, nous traduirons le terme anglais « *smart* » par le terme « *d'intelligence intégrée* ». Cette traduction synthétise bien les précédentes définitions. Cependant pour des raisons de commodité, nous continuerons à employer l'expression « *smart* ».

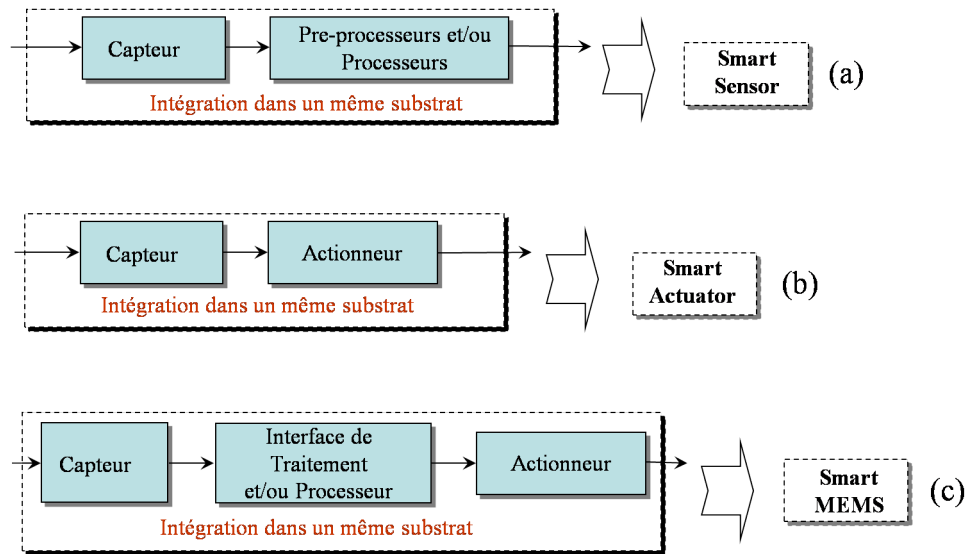


Fig. 1.12 – Représentation de dispositifs type « smart ». (a) « smart sensor ». (b) « smart actuator ». (c) « smart MEMS ».

1.3.3 Exemples de « Smart Structure »

Un exemple classique d'une « smart structure » est celle d'un bâtiment qui contiendrait un certain nombre de capteurs de mouvement de structure avec un système d'amortissement actif. La structure du bâtiment peut alors répondre aux modifications de son environnement (ex : fort vent) et modifier sa réponse mécanique de façon appropriée (ex : à travers son coefficient d'amortissement variable).

Une « smart structure » peut aussi être utilisée pour tester la santé de certaines habitacles de type industriel. Par exemple, l'état d'étanchéité ou de solidité d'une structure d'avion. Détecter les problèmes dès le début ou adapter la réponse de manière à réagir à des conditions imprévues permettrait d'augmenter la viabilité de la structure et d'en améliorer le cycle de vie.

Il apparaît aussi dans ces applications que la notion de « contrôle distribué » est hautement importante à l'amélioration des performances de ces produits. Ainsi, pour améliorer les conditions d'exploitation de certaines « smart structures », il faudra réaliser des progrès notables dans le domaine du contrôle et du « contrôle distribué » plus précisément.

Nous introduirons donc dans la section suivante quelques notions de « contrôle distribué » et l'état de la recherche dans ce domaine appliqué aux MEMS et aux « smart structures ».

1.4 Contrôle distribué

1.4.1 Topologie

1.4.1.1 L'approche de Berlin et Gabriel

A l'époque *Berlin et Gabriel* mettaient en avant, comme principal défi « informatique » du « *contrôle distribué* », la connexion (dans le sens de couplage) étroite qui existait entre les dispositifs à MEMS et l'environnement avec lequel ils interagissaient. Ce couplage entraînerait d'après eux que le « calcul » soit dirigé par des événements dans l'environnement en temps réel, rendant incontournables les équations différentielles, qui modélisent le comportement du monde physique, comme une part implicite d'un programme informatique distribué. Les questions de « où et quand détecter ? », « comment les lectures de plusieurs capteurs différents doivent être corrélées pour déterminer quelle action prendre ? » et « où et quand une action va avoir l'effet désiré ? », sont toutes déterminées par la physique de l'environnement associé à chaque application. La réponse à ces questions peut varier autant de fois que l'environnement changera.

Pour définir la stratégie de « *contrôle distribué* », il faut d'abord connaître le type d'application à laquelle elle se prédestine. *Berlin et Gabriel* avaient établi trois différentes classes d'applications du « *contrôle distribué* » :

- Les « *smart particles* » distribuées dans un environnement, dans lequel l'emplacement relatif des dispositifs alterne d'un lieu à un autre au cours du temps. Les défis propres aux « *smart particles* » incluent une détermination des emplacements relatifs, un établissement d'un réseau de communication variant dans le temps (*time-varying*), et d'action accomplie en collaboration et synchronisation, même pour des distances étendues. Un exemple très connu de ces « *smart particles* » sont les « MEMS dust » (« poussière de *MEMS* ») [30].
- Les « *surfaces actives* » (*active surfaces*), dans lesquels les dispositifs sont en permanence attachés à une surface suivant une topologie établie. Les dispositifs sont couplés d'abord pour participer à la dynamique du médium qu'ils sont en train de manipuler, et non pour la dynamique de la surface à laquelle ils sont attachés. Le premier défi de la technologie de l'information posé par les « *surfaces actives* » est d'impliquer dynamiquement les voisins de chaque dispositif à travailler ensemble pour interagir avec le monde physique à une échelle locale, qu'il s'agisse de positionner une pièce pour l'assemblage ou d'influencer un « vortex » sur une aile d'avion.
- Les « *smart structures* », dans lesquelles les éléments MEMS sont arrangés et leurs interactions couplées à travers la dynamique du matériau auquel ils sont attachés, menant à un besoin de coordination aussi bien au niveau global (centralisé) qu'au niveau local (décentralisé).

Parmi les diverses configurations de MEMS, chacune pose un ensemble quelque peu différent de défis « informatiques », lesquels peuvent être classés par les facteurs listés sur la figure 1.13. Par exemple, dans le cas de la surveillance de particules en suspension, une part du défi est la

configuration spatiale variable dans le temps (topologie dynamique).

Pour les « surfaces actives » aérodynamiques, le contrôle est compliqué par les dynamiques du fluide en train d'être manipulé, et par le besoin pour les actionneurs d'assumer différents rôles dépendant de la localisation (organisation logique variable, se connectant avec le monde physique).

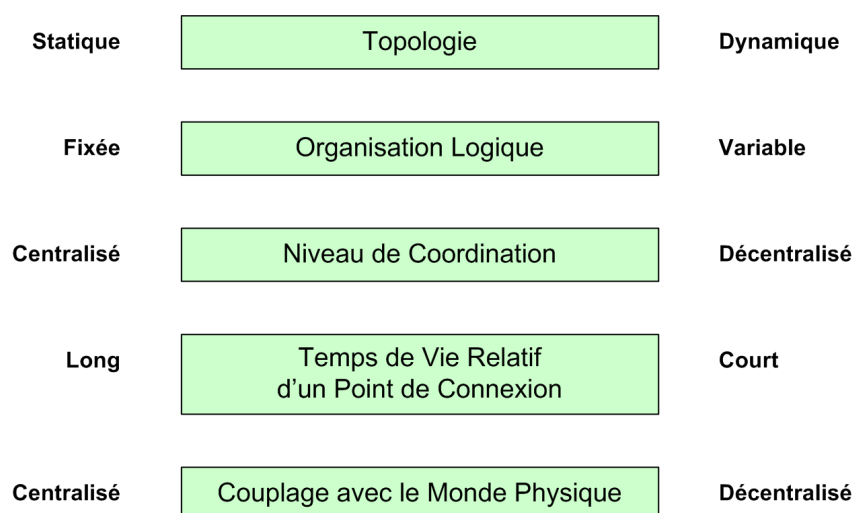


Fig. 1.13 – Classification des facteurs topologiques du « contrôle distribué ».

On remarque dans cette approche topologique du « contrôle distribué », l'importance des notions de contrôle *centralisé* et *décentralisé*. Nous allons détailler ces principes dans ce qui suit.

1.4.1.2 Contrôle centralisé et/ou décentralisé

Généralement, un système à contrôle centralisé consiste en une série de microsystemes qui se composent uniquement d'un capteur et d'un microactionneur. La stratégie de contrôle du système est effectuée par un contrôleur central. Par conséquent, tous les capteurs et tous les actionneurs doivent être connectés au contrôleur central, ce qui augmente considérablement le nombre de connexions. Le coût et la complexité du système augmentent également, alors que la fiabilité du système diminue. La figure 1.14(a) montre la topologie générale d'un système à contrôle centralisé.

En revanche, dans un système à contrôle décentralisé, tous les microsystemes composés d'un capteur et d'un actionneur possèdent aussi leur propre partie intelligente (contrôleur), ils n'ont donc pas besoin de contrôle central ou de communication globale. Dans un système à contrôle décentralisé, la communication et le contrôle se passent localement, les informations s'échangent entre les voisins immédiats. Par conséquent, il n'y a pas de décision globale ou centrale qui est prise pour le contrôle du fonctionnement du système. Pourtant, le système est plus robuste et tolérant en cas de défaillance du contrôle du système. En effet, étant donné que chaque élément de base du système contribue au contrôle du système global, le mauvais fonctionnement d'une

partie d'entre eux pourrait ne pas entraîner l'échec total du système, ce qui n'est pas le cas pour le système à contrôle centralisé qui ne possède qu'une unité de contrôle. La figure 1.14(b) montre la topologie générale d'un système à contrôle décentralisé.

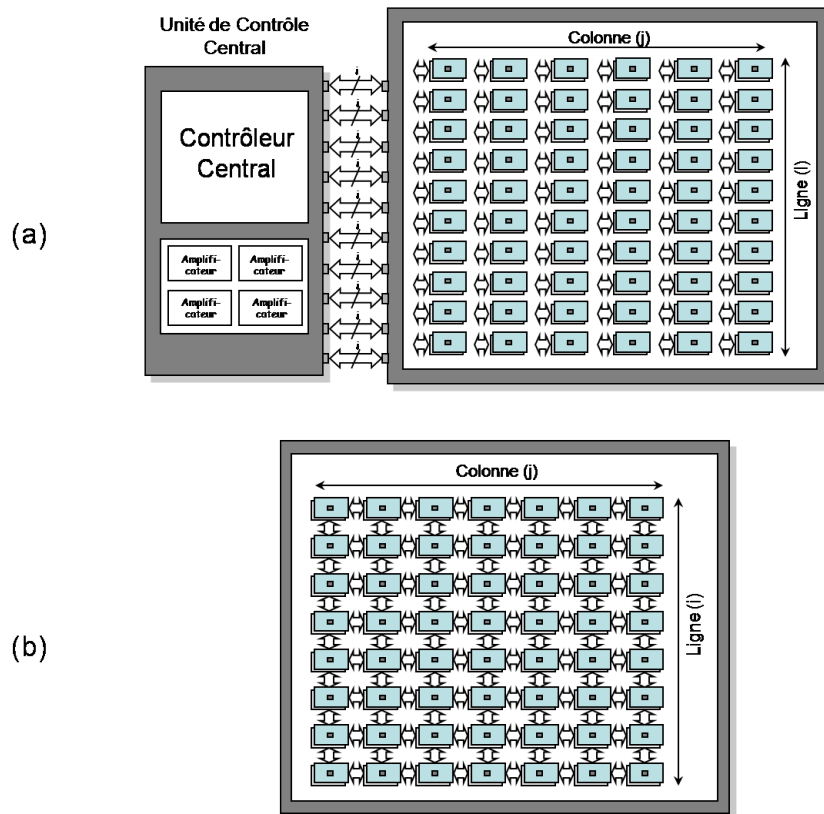


Fig. 1.14 – *Approche de contrôle centralisé et décentralisé. (a) Centralisé. (b) Décentralisé.*

1.4.1.3 Contrôle de système distribué

Le système distribué est similaire au système à contrôle décentralisé, mais doté aussi de caractéristiques appartenant au système à contrôle centralisé. C'est donc une solution hybride centralisé/décentralisé qui peut souvent faciliter l'approche de conception et d'implantation de telles structures de contrôle.

Pour chaque microsystème, le capteur et l'actionneur sont reliés au contrôleur local, qui sont reliés à leur tour aux autres contrôleurs qui pourraient être des contrôleurs locaux ou un contrôleur central. La communication entre les contrôleurs locaux et le contrôleur central peut se réaliser au moyen d'un protocole de haut niveau à travers un seul fil. Un tel système est montré par la figure 1.15.

De manière plus générale, nous parlerons d'agent ou de systèmes multi-agents lorsque nous aborderons ces structures de contrôle *décentralisé* ou *distribué*, comme nous les avons introduits précédemment. L'agent peut être interprété comme l'entité physique ou virtuelle d'un composant

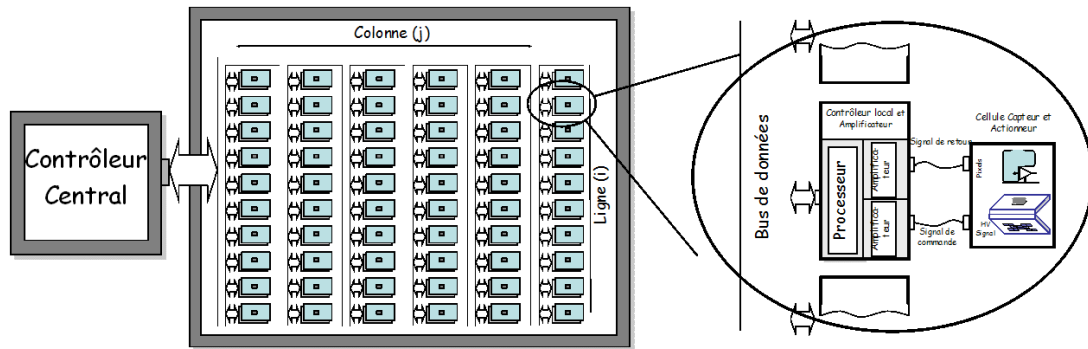


Fig. 1.15 – Approche de contrôle distribué

MEMS dans les « *smart structure* » que nous étudions. Voyons plus en détail ces notions.

1.4.2 Notions d'agent et multi-agents

1.4.2.1 Contexte

Depuis une dizaine d'année, les systèmes multi-agents ont connu un grand essor et sont appliqués à des domaines très variés comme, par exemple, pour la simulation de la vie artificielle, pour la robotique ou le traitement d'images [31]. Les systèmes multi-agents sont issus de l'*intelligence artificielle distribuée* (IAD), une branche de l'intelligence qui s'articule autour de trois axes :

- *La résolution distribuée* des problèmes qui s'intéresse à la manière de diviser un problème en un ensemble d'entités distribuées et coopérantes et à la manière de partager la connaissance du problème afin d'en obtenir la solution.
- *L'intelligence artificielle* parallèle qui développe des langages et des algorithmes parallèles pour l'intelligence artificielle (IA) visant ainsi l'amélioration des performances des systèmes d'IA.
- *Les systèmes multi-agents* qui privilégient une approche « *décentralisée* » de la modélisation et mettent l'accent sur les aspects collectifs des systèmes. Les systèmes multi-agents, comme leur nom l'indique, sont des systèmes constitués de plusieurs agents. Il est donc nécessaire de commencer par définir ce que l'on appelle un agent. La notion d'agent définie, nous introduirons le concept de systèmes multi-agents.

1.4.2.2 L'agent

Il n'existe pas, actuellement, une définition de l'agent qui fasse l'unanimité dans le monde de l'intelligence artificielle distribuée. Cependant, nous nous baserons sur celle qu'a introduite *J. Ferber* car elle nous paraît la mieux adaptée à notre étude [32].

J. Ferber définit un agent comme étant une entité physique ou virtuelle évoluant dans un environnement dont il n'a qu'une représentation partielle (*perception + raisonnement*) et sur

lequel il peut agir (*raisonnement + agir*), comme le montre la figure 1.16. Il est aussi capable de communiquer avec d'autres agents et est doté d'un comportement autonome. Cette définition aborde une notion essentielle : l'**autonomie**. En effet, ce concept est au centre de la problématique des agents. L'autonomie est la faculté d'avoir ou non le contrôle de son comportement sans l'intervention d'autres agents. Une autre notion importante abordée par cette définition concerne la capacité d'un agent à communiquer avec d'autres.

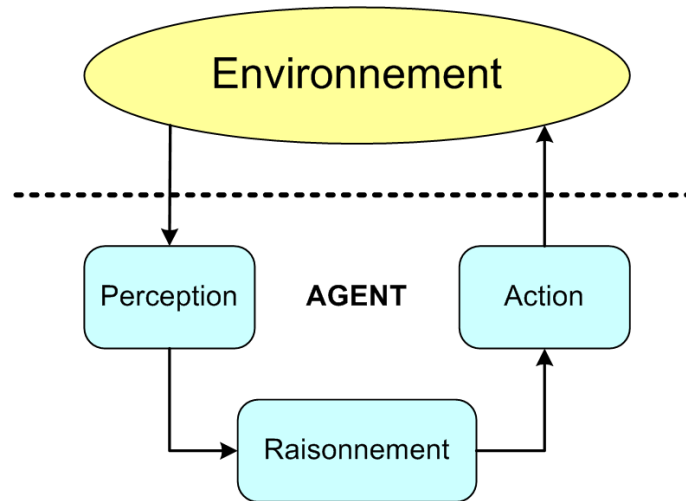


Fig. 1.16 – *Fonctionnement d'un agent par rapport à son environnement*

1.4.2.3 Le système multi-agents

Un système multi-agents est un ensemble d'agents qui évoluent dans un environnement commun. On peut définir l'intelligence artificielle distribuée comme étant l'étude, la conception et la réalisation de systèmes multi-agents qu'il présente comme étant des systèmes dans lesquels des agents intelligents interagissent et poursuivent un ensemble de buts ou réalisent un ensemble d'actions.

On peut donc se demander comment, à partir d'un ensemble d'agents pas forcément intelligents, on arrive à un système complexe où semble se dégager de l'intelligence. Dans un système multi-agents l'intelligence provient de l'émergence d'un comportement global.

1.4.2.4 L'environnement des systèmes multi-agents

Dans un système multi-agents, on appelle environnement l'espace commun aux agents du système. Un environnement peut être défini comme :

- Accessible si un agent peut, à l'aide des primitives de perception, déterminer l'état de l'environnement et ainsi procéder, par exemple, à une action. Si l'environnement est inaccessible alors il faut que l'agent soit doté de moyens de mémorisation afin d'enregistrer les modifications qui sont intervenues.

- Déterministe ou non, selon que l'état futur de l'environnement soit, ou non, fixé que par son état courant et les actions de l'agent.
- Épisodique si le prochain état de l'environnement ne dépend pas des actions réalisées par les agents.
- Statique si l'état de l'environnement est stable (ne change pas) pendant que l'agent réfléchit. Dans le cas contraire, il sera qualifié de dynamique.
- Discret si le nombre des actions faisables et des états de l'environnement est fini. Cette partie de l'analyse consiste donc à trouver les éléments nécessaires à la réalisation des interactions extérieures au système comme par exemple la perception de cet environnement, les actions que l'on peut y faire.

En abordant ces notions liées à l'« *environnement des systèmes multi-agents* », on montre bien toute la difficulté à modéliser ou simuler des structures de « *contrôle distribué* » dans un environnement multi-physique et multi-technologie. C'est toute l'importance de cette thèse et de nos travaux de recherche sur la simulation multi-domaine dont nous introduisons les notions dans la section suivante.

1.5 Simulation multi-domaines

1.5.1 Approche de modélisation

Aujourd'hui, les concepteurs ont la possibilité d'améliorer les simulations au niveau système en intégrant des modèles plus réalistes des composants électroniques. Les défauts de conception et d'intégration sont détectés plus tôt dans le processus de conception. Les concepteurs de systèmes complexes à base de MEMS doivent donc effectuer des simulations pertinentes pour obtenir des conceptions justes du premier coup. Ces simulations pour être fiables nécessitent une modélisation fine des composants mais également la prise en compte des interactions entre les différents éléments du système et de son environnement.

Les phénomènes dans les composants MEMS sont par nature multi-domaines (multi-physiques et multi-technologiques). En effet, aux comportements principaux qui mêlent la mécanique, l'électronique, s'ajoutent plusieurs effets physiques comme la chaleur, les ondes et champs (électrostatique, électromagnétique, etc.), les fluides (liquides, écoulement d'air, etc.).

Comment décrire de tels modèles multi-domaines en vue de leur simulation et leur réalisation industrielle? Nous présentons ici trois types de modélisations qui nous semblent les plus représentatifs de la démarche de modélisation des concepteurs :

- *La modélisation par éléments finis,*
- *La modélisation analytique,*
- *La modélisation descriptive.*

1.5.2 La modélisation par éléments finis

La méthode des *éléments finis* fait partie des outils de mathématique appliquée. Il s'agit de mettre en place, à l'aide des principes hérités de la formulation variationnelle ou formulation faible, un algorithme discret mathématique permettant de résoudre une *équation aux dérivées partielles* ou EDP sur un domaine compact avec conditions aux bornes et/ou dans l'intérieur du compact. On parle couramment de conditions de type Dirichlet (valeurs aux bornes) ou Neumann (normales aux bornes) [33].

Elle est très utilisée dans l'industrie mécanique, mécanique des fluides, électromagnétique, acoustique, thermique, en particulier en aéronautique, dans l'industrie automobile, en météorologie, etc. La plupart du temps cette méthode ne donne accès qu'à l'étude d'un seul domaine physique à la fois. Cette méthode est intéressante, compte tenu de sa souplesse d'utilisation, en particulier vis-à-vis de l'approximation des divers opérateurs modélisant des phénomènes en physique-mathématique et également pour la prise en compte de conditions aux limites portant sur les gradients de la fonction à calculer. Pour référence, nous citerons les outils commerciaux comme ANSYS®, Abaqus® ou encore COMSOL Multiphysics®.

Sur le plan informatique, la méthode des éléments finis conduit à l'écriture de code de calculs les plus généraux possible, ce qui correspond certes à un avantage mais aussi à un inconvénient, compte tenu de la difficulté pratique de programmation de cet algorithme ; il convient de noter cependant que le schéma de principe du code est relativement simple, la complexité découlant des innombrables possibilités qu'offre la méthode. De plus, le développement d'un tel code nécessite de longs mois de programmation [34]. D'ailleurs les codes généraux sont optimisés spécifiquement à chaque domaine physique, voire à chaque application.

Une autre difficulté de compréhension de la méthode des éléments finis réside dans le formalisme mathématique préalable et sous-jacent à la mise en oeuvre algorithmique. En effet, compte tenu de la complexité croissante des modèles mathématiques permettant la compréhension de phénomènes de plus en plus compliqués à expliquer, il a été nécessaire de s'appuyer sur des résultats d'analyse fonctionnelle élaborés pour formuler cette méthode d'approximation.

Il faut aussi noter qu'il est impossible d'aborder par cette méthode un système au niveau global aussi bien par la spécificité de la modélisation (les EDP), que par la non gestion des discontinuités que par le volume extrêmement important de calcul à mener.

Ces techniques pour puissantes qu'elles soient font partie des outils dits « métiers ».

1.5.3 Modélisation analytique

Pour résoudre ce problème de temps de calcul très long, des modélisations dites « *analytiques* » peuvent être envisagées.

Il s'agit ici, moyennant quelques simplifications, d'utiliser directement les équations mathématiques représentant le comportement des systèmes dans les différents domaines physiques. On

privilégie ici, quand c'est possible ou dans un cas d'espèce, la formulation globale plutôt que l'expression locale du comportement en remplaçant des équations aux dérivées partielles par des équations différentielles. Pour résoudre ces équations, des outils de résolution mathématique, tels que Matlab, sont utilisés. Par rapport aux modélisations aux éléments finis, ce nouveau type de modélisation permet l'obtention beaucoup plus rapide des résultats au détriment de leur résolution spatiale.

La réduction de l'ordre des modèles est l'une des approches les plus couramment utilisée dans le domaine de la simulation, du contrôle et de l'optimisation des processus physiques complexes [35]. La réduction de l'ordre du modèle est souvent cruciale pour accélérer la simulation de tels systèmes à grande échelle, comme des structures à base de MEMS. En fait, les systèmes peuvent être si grands que les simulations ou même le stockage des données décrivant de tels systèmes sont prohibés sans remplacer en premier et de façon appropriée les systèmes à grande échelle originaux par des modèles à ordre réduit de dimension très nettement inférieure. Ceci est encore plus évident si non seulement la simulation du processus physique est requise mais également si le contrôle ou l'optimisation est l'objectif de l'étude.

En dépit de ces progrès, il reste de nombreux problèmes en suspens. Par exemple, la réduction de l'ordre des systèmes non-linéaires à grande échelle est encore à l'état embryonnaire. Également, de nombreuses méthodes de réduction, qui ont été conçues ces dernières années, sont décrites dans les termes utilisés dans leur discipline respective ou même des applications spécifiques, et cela même s'ils partagent de nombreuses caractéristiques et origines.

1.5.4 Modélisation descriptive

Le troisième type de modélisation multi-domaines est en train de connaître le plus grand succès, il s'agit de la modélisation descriptive, et elle s'appuie sur des langages dédiés à la modélisation multi-domaines.

Ces langages permettent non seulement de développer des modèles de systèmes électriques mixtes (analogique et numérique), mais aussi des systèmes non-électriques (mécanique, magnétique, thermique, etc.) par l'utilisation d'une description de texte. Parmi ces langages les plus utilisés, nous pouvons citer le langage MAST du simulateur SABER, le langage Verilog-AMS, ainsi que le langage VHDL-AMS. A part MAST qui est disponible uniquement avec le simulateur SABER, Verilog-AMS et VHDL-AMS sont beaucoup plus accessibles grâce aux produits de Synopsys®, Cadence®, Ansoft Corporation® ou de Dolphin Integration®. Le seul langage normalisé est le VHDL-AMS.

L'utilisation des langages présente de nombreux avantages, comme la portabilité, la réutilisabilité par rapport aux deux autres types de modélisation. En plus, avec le développement des outils de simulation, nous pouvons effectuer des co-simulations dans un seul simulateur, c'est-à-dire de simuler des modèles combinant des modules développés en différents langages. Par exemple, depuis 2005, Synopsys® a commencé les études sur la combinaison de MAST et

VHDL-AMS dans un modèle. Si cette tendance se confirme, les utilisateurs vont en tirer beaucoup de bénéfices. Car les IP (Intellectual Property) accumulés dans de nombreuses bibliothèques développées depuis des années deviennent tellement grandes qu'il faudrait dépenser énormément d'effort pour les récrire avec un autre langage.

Depuis sa normalisation en 1999, VHDL-AMS a accéléré sa conquête dans le secteur de la modélisation multi-domaines.

Le langage de description de matériel VHDL-AMS, extension aux systèmes à temps continu et mixtes du VHDL, spécifique à la description de systèmes à temps discret, est spécialement conçu pour le multi-domaines [36]. Il permet de décrire et de simuler les systèmes multi-technologiques grâce à la spécification :

- de terminaux dont les grandeurs physiques associées respectent implicitement les lois de Kirchhoff généralisées quand ils sont connectés entre eux. Par exemple, les échanges thermiques conductifs se modélisent alors simplement par la déclaration de terminaux thermiques auxquels sont associées deux quantités : la température et la puissance thermique.
- de quantités analogiques dont les natures physiques peuvent être quelconques. Le comportement des composants multi-domaine se traduit alors par un jeu d'équations différentielles non linéaires couplées impliquant des quantités de natures différentes.

L'écriture d'un modèle en VHDL-AMS est directe et la difficulté de résolution des équations est transparente pour le programmeur. En effet, l'écriture d'une équation physique complexe - équation différentielle non linéaire implicite ou explicite à discontinuité éventuelle - ne nécessite qu'une instruction simultanée, soit une ligne de code en VHDL-AMS. Puis la résolution est assurée par les puissants simulateurs à temps continu développés pour la CAO électronique depuis plus de 40 ans. De plus les équations différentielles ne peuvent être valables que sur un domaine fini du temps. Le changement d'équations descriptives et les discontinuités associées sont gérables par le langage par l'intermédiaire d'instructions simples et de mécanismes bien contrôlés.

Le modèle VHDL-AMS du composant permet alors de le simuler dans un outil de conception électronique - conception d'ASIC mixte - avec des circuits de pré ou de post traitement analogiques et numériques tout en tenant compte de l'environnement effectif de son utilisation. Deux avantages supplémentaires du VHDL-AMS servent les intérêts des industriels :

- le langage est normalisé (IEEE 1076.1 1999). Les modèles sont donc indépendants des outils de simulation et portables. Ceci facilite leur réutilisation.
- le langage VHDL-AMS permet la multi-abstraction des modèles : des modèles plus ou moins détaillés (fonctionnels, comportementaux, ou physiques) peuvent être décrits, mixés et simulés. Ceci facilite la conception hiérarchique des systèmes complexes.

De par ces propriétés, le langage VHDL-AMS s'avère parfaitement adapté à la problématique de notre thèse. Nous aborderons donc nos modèles de simulation multi-domaines par cette description. Nous reviendrons plus tard, dans le Chapitre 3 de ce mémoire, sur l'environnement de conception de ce langage (méthodologie, outils, etc.).

1.6 Conclusion

Ce chapitre d'introduction à la problématique de notre thèse, nous a permis de prendre toute la dimension des défis technologiques restant à relever aujourd'hui dans le domaine de la conception de « smart structures » à base de MEMS. Domaine qui est certainement amené à générer des applications à haute technologie très prometteuses dans le futur.

Plus spécifiquement, nous avons pu mettre en avant l'importance des procédés de « contrôle distribué » et de « simulation multi-domaines » dans la réalisation de telles structures complexes. En effet, à notre connaissance, peu ou pas de recherche ont encore pu faire émerger des résultats impliquant l'ensemble des éléments inhérents à cette problématique de conception.

A travers ces premières introductions, nous avons pu dessiner les contours de notre thèse qui se veut une contribution au domaine de conception par « simulation multi-domaine » à partir du langage VHDL-AMS qui a spécialement été conçu pour la description de ce type de modèles multi-physique et multi-technologique.

Notre démarche de thèse, nous a amené à travailler sur la description d'une « smart structure » type « smart surface » qui est un dispositif de micromanipulation « sans contact » basé sur l'intelligence et l'autonomie de microactionneurs pneumatiques pour la génération des jets d'air distribués.

Dans le Chapitre 2, nous aborderons l'état de l'art de ce type de « smart surface » et le prototype microfluidique que nous avons choisi de contrôler, modéliser et simuler.

Chapitre 2

« Smart Surface » à base de MEMS : État de l'art et prototype pneumatique

2.1 État de l'art

2.1.1 Introduction

Pour valider notre approche de modélisation et simulation multi-domaines, le choix d'une application type « *smart structure* » à base de MEMS a été pensé en fonction des contraintes de conception et de fabrication inhérentes à ce type de dispositif. Nous souhaitons modéliser une structure à la fois multi-technologique (MEMS, composant intégré, assemblage 3-D, systèmes distribués, etc.) et multi-physique (mécanique, électrique, etc.). Il fallait aussi que ce dispositif ait déjà été réalisé et validé expérimentalement pour permettre une comparaison réaliste de nos futurs résultats de simulation.

Ainsi, nous avons été amenés à collaborer avec des chercheurs du domaine des MEMS et de leurs applications. Ceci a été rendu possible par les recherches de *Y.-A. Chapuis* qui à l'époque développait des dispositifs de « surface active » à base de MEMS au sein du groupe du *Pr. Fujita* de l'Université de Tokyo au Japon. Plus particulièrement, nous nous sommes intéressés à un prototype développé à partir de microactionneurs pneumatiques distribués générant des forces fluidiques pour la micromanipulation 2-D d'objets de taille millimétrique [37].

Le domaine d'application de ce type de dispositif pourrait être le secteur industriel des micro- et nano-usinages. En effet, les dimensions impliquées par des dispositifs type « *smart structure* » à base de MEMS privilégient naturellement les applications aux échelles millimétrique et/ou micrométrique et concernent, notamment, la manipulation et l'hybridation des composants sur silicium dans l'industrie électronique [20]. Ces réseaux de microactionneurs pourraient même répondre à certaines applications de puissance à l'échelle macroscopique, les investigations les

plus récentes ayant démontré la faisabilité de réseaux matriciels permettant de développer des densités de force pouvant atteindre $0,5 \text{ N/mm}^2$.

Le volume consacré à ce chapitre ne permet naturellement pas de recenser l'ensemble des applications pressenties dans ce domaine. Nos investigations concerneront prioritairement les « *surfaces actives* » à base de MEMS destinées à la « *manipulation parallèle massive* » de milli- et microcomposants comme celles impliquées dans les secteurs-clés du développement futur des micro- et nano-systèmes.

2.1.2 Concepts de micromanipulation distribuée

2.1.2.1 Principes de manipulation distribuée

Le développement de « *surfaces actives* » à base de MEMS, capables de répondre aux enjeux de la manipulation parallèle et massive de microcomposants, impose l'investigation de nouveaux systèmes distribués présentant une architecture de type « mosaïque ». Le principe consiste à associer un grand nombre de cellules de forces élémentaires mécaniquement découplées dans le but de discrétiser les forces utiles élémentaires agissant à la surface du manipulateur. On parlera alors de « *manipulation distribuée* » [38].

L'idée se base sur l'imitation des concepts de manipulation d'objets à taille macro où un objet peut être déplacé en répartissant des forces de manutention sur différents points de contact d'une zone ou d'une surface. Ces principes sont illustrés sur l'image de la figure 2.1 où l'on observe la manipulation d'une personne au-dessus de la foule. Chaque personne de la foule peut être considérée comme un système intelligent et autonome, permettant une coopération interactive de manipulation de la personne déplacée.



Fig. 2.1 – Concept de « *manipulation distribuée* » (Photo extraite de l'ouvrage « *Distributed manipulation* » [38])

Transposer cette idée jusqu'au concept de « *micromanipulation distribuée* » était jusqu'à encore récemment impossible. La technologie ne permettait pas de satisfaire les contraintes inhérentes à la construction de telles architectures mais les récentes évolutions enregistrées dans le domaine de la microfabrication ont permis de reconsidérer en profondeur le concept des micromanipulateurs industriels.

2.1.2.2 Vers la micromanipulation distribuée

Comme le reporte *P. Minotti* dans une synthèse sur les « *réseaux de microactionneurs employés en micromanipulation* », l'infrastructure industrielle actuelle, composée essentiellement d'équipements macroscopiques lourds, lents, encombrants et peu précis, ne saura répondre plus longtemps aux exigences ultimes de la miniaturisation [20]. En effet, les manipulateurs macroscopiques conventionnels, centrés par tradition sur la préhension unitaire et sérielle de composants macroscopiques au moyen de robots « *pick and place* », ne peuvent répondre efficacement aux enjeux de l'assemblage en masse des composants microscopiques.

Pour *P. Minotti*, l'échelle et la densité des composants aujourd'hui réalisés à partir de la mise en oeuvre des microtechnologies justifient une complète remise en cause des méthodologies classiques de préhension et d'assemblage. Selon toute vraisemblance, les micromanipulateurs du futur devront donc évoluer vers des dispositifs matriciels ou distribués impliquant des « *surfaces actives* » permettant d'assurer une manipulation parallèle massive, centrée notamment sur la gestion et le contrôle en temps réel de champs de forces locaux et multidirectionnels.

La figure 2.2 illustre l'exemple d'une « *surface active* » à base de microsystèmes distribués créant des champs de vecteurs (champ de forces) pour la micromanipulation 2-D [39]. Deux cas de micromanipulation sont présentés (translation et rotation).

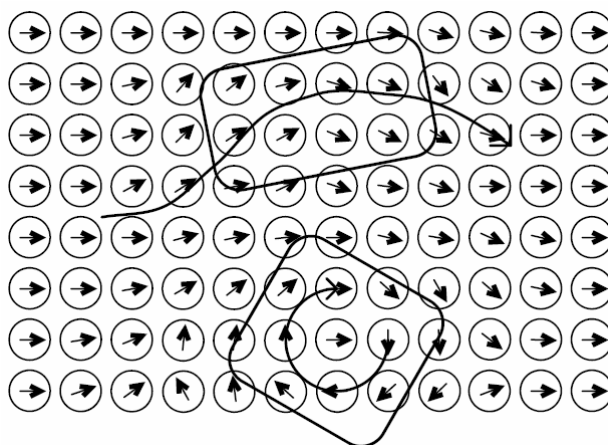


Fig. 2.2 – Exemples de « *surface active* » pour micromanipulation matricielle ou distribuée (image extraite de l'article [39])

Le moment est donc venu de reconsidérer l'échelle des systèmes de production, en adaptant celle-ci aux dimensions des composants impliqués dans les processus de fabrication des micro-

systèmes. Cette nécessaire adaptation d'échelle implique une remise en cause globale de l'architecture mécanique des systèmes de manipulation actuels, en prenant appui sur une technologie de fabrication pertinente à l'échelle des microsystèmes.

Les capacités actuelles des premiers réseaux de microactionneurs plaident largement en faveur des manipulateurs matriciels à base de MEMS. Ces dispositifs pourraient prochainement supplanter les manipulateurs macroscopiques conventionnels dans le domaine de la manipulation parallèle des composants industriels. Les pionniers dans ce domaine d'activité ont été *K. Pister* dès 1990 [40], *H. Fujita* entre 1993-1995 [41] et *K. Böhringer* vers 1996 [42]. Depuis, ces recherches se sont poursuivies avec succès pour de nombreux cas de réalisation, comme ceux que nous allons décrire dans ce qui suit.

2.1.3 Technologies des micromanipulateurs distribués à base de MEMS

Les « *surfaces actives* » intégrant des densités importantes de modules actifs par unité de surface sont par conséquent l'illustration type des recherches actuellement conduites dans le domaine des micromanipulateurs en technologie MEMS. On peut résumer les stratégies de recherche actuelles par deux approches de conception :

1. Pour satisfaire le convoyage d'objets éventuellement macroscopiques, il faut en premier lieu répartir un grand nombre de microactionneurs à la surface d'un plan de transfert généralement horizontal. On exploite ainsi le poids propre des objets transportés, pour calibrer la précontrainte extérieure à l'origine du transfert d'énergie mécanique sur les éléments extérieurs à la surface. Une telle configuration permet aussi de minimiser le travail mécanique extérieur requis pour manipuler les objets à la verticale du manipulateur. On pourra distinguer deux types de « surface active » ou micromanipulateur 2-D :
 - *Systèmes « à contact » (contact systems),*
 - *Systèmes « sans contact » (contact-free systems).*
2. Il est ensuite nécessaire d'associer des capteurs et des unités de contrôle à chaque microactionneur de la surface active 2-D afin de maîtriser les déplacements des objets manipulés. Les actionneurs, capteurs et circuits de contrôle sont intégrés dans des servosystèmes primaires, encore appelés modules. On parlera dans ce cas de « *smart structure* » ou « *smart surface* ».

2.1.3.1 Micromanipulateurs 2-D « à contact » et « sans contact »

a. Micromanipulateurs « à contact »

Les systèmes « à contact » représentent la catégorie la plus importante des micromanipulateurs 2-D à base de MEMS distribués. Ils se définissent comme un réseau de microactionneurs dont chaque unité vient *en contact* avec l'objet qu'il est supposé manipuler.

On prendra l'exemple de la « surface active » à base de microactionneurs en polyimide dont l'effet thermique permet de déformer et déplacer verticalement des pieds mobiles de manière à créer l'actionnement de contact avec l'objet à convoier (Cf. figure 2.3(a)). Ce dispositif a été développé dès 1993 par *M. Ataka* au sein du groupe du *Pr. Fujita* de l'Université de Tokyo au Japon [43].

D'autres cas de réussite de systèmes « à contact » peuvent être cités dans cette introduction comme les « surfaces actives » à base de microactionneurs électrostatiques [42]. Nous reviendrons sur les principes et performances de ces dispositifs plus loin dans ce chapitre.

La figure 2.3 montre la réalisation des concepts de micromanipulateur 2-D « à contact » mentionnés précédemment.

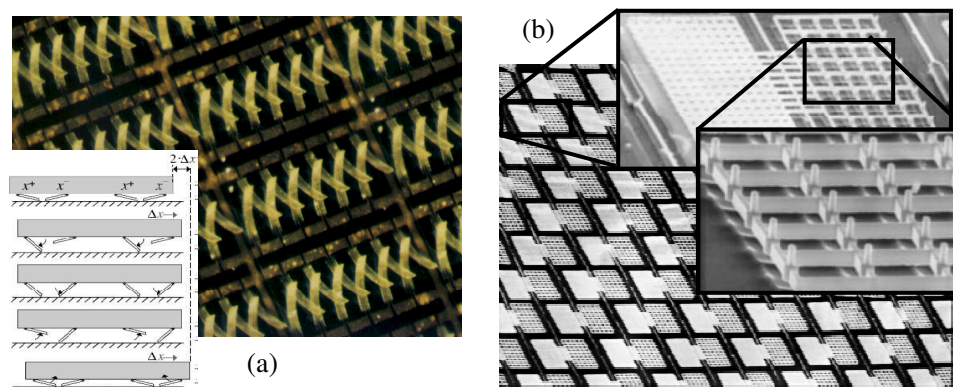


Fig. 2.3 – Exemples de technologie de micromanipulateur 2-D « à contact ». (a) Par pieds mobiles par microactionnement (image extraite de l'article [43]). (b) Par microactionneurs électrostatiques (image extraite de l'article [42])

b. Micromanipulateurs 2-D « sans contact »

Les systèmes « sans contact » se définissent comme un réseau de microactionneurs dont chaque unité produit une force à l'interface surface/objet, évitant ainsi tout contact ou frottement mécanique de la surface avec l'objet manipulé.

On prendra l'exemple de la « surface active » à base de microactionneurs pneumatiques dont l'actionnement d'une micro-valve permet le passage d'un écoulement d'air ou de gaz de manière à créer un jet d'air vertical et horizontal capable de maintenir en lévitation et de pousser l'objet à convoier. Le premier dispositif à base de microactionneurs pneumatiques a été développé dès 1990 par *K. Pister* au sein du groupe du *Pr. Howe* de l'Université de Berkeley en Californie (États-Unis) [40]. D'autres réalisations suivront comme celles développées par *S. Konishi* au sein du groupe du *Pr. Fujita* de l'Université de Tokyo au Japon [44].

Un second cas de systèmes « sans contact » est aussi souvent cité en exemple pour ses performances de micromanipulation 2-D, il s'agit des systèmes à base de microactionneurs électromagnétiques [45]. Nous reviendrons sur les principes et performances de ces systèmes un peu plus loin dans ce chapitre.

La figure 2.4 montre la réalisation des deux dispositifs de micromanipulateur 2-D « sans contact » mentionnés précédemment.

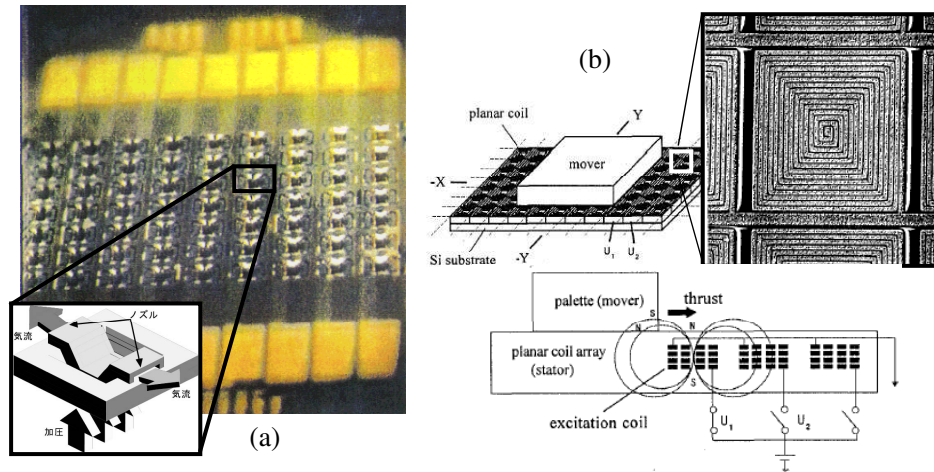


Fig. 2.4 – Exemples de technologie de micromanipulateur 2-D « sans contact ». (a) Par microactionneurs pneumatiques (images extraites de l'article [44]). (b) Par microactionneurs électromagnétiques (images extraites de l'article [45])

c. Comparaison des performances de micromanipulation 2-D

A travers la littérature récente et plus ancienne, il a été possible d'établir un tableau de comparaison de performances entre les principaux types de micromanipulateurs 2-D « à contact » ou « sans contact », tel que l'ont réalisé en 2000 *T. Ebefors* et *G. Stemme* dans l'ouvrage « *The MEMS Handbook* » [46]. Cette synthèse reprise dans le tableau 2.1 met en évidence les avantages et les inconvénients des solutions dites « à contact » et celles dites « sans contact ».

Finalement, les solutions « à contact » présentent de meilleurs aspects en ce qui concerne la performance de précision ou le coût de fabrication. Pourtant, les inconvénients tels que la fragilité et la faible vélocité de tels dispositifs ne plaident pas entièrement en leur faveur pour les applications du futur.

En revanche, dotés d'avantages incomparables par rapport aux solutions « à contact », les solutions « sans contact » ouvrent de nouvelles perspectives d'applications. Cependant ces systèmes doivent réduire leur coût de fabrication encore trop élevé et améliorer leurs performances de précision.

2.1.3.2 Principes de « smart surfaces » à base de MEMS

Si les premiers prototypes de micromanipulateurs 2-D ont permis d'atteindre d'excellentes performances de fonctionnement, ces dispositifs ne répondent pas encore totalement aux exigences de stabilité et de précision de contrôle requises pour la manipulation parallèle de micro-composants industriels.

L'intégration d'éléments de contrôle et de capteur sur un composant MEMS est un facteur

Micro- actionneurs Distribués	Systèmes « à Contact »		Systèmes « sans Contact »	
	Microactionneurs Electrostatiques [42, 47]	Pieds mobiles par Microactionnement [43, 48]	Microactionneurs Pneumatiques [40, 44]	Microactionneurs Electromagnétiques [45]
<i>Principes</i>	Un réseau de microactionneurs électrostatiques contrôlés pour produire une force mécanique.	Un réseau de pieds mobiles élevés par effet physique pour produire une force mécanique.	Un réseau de microactionneurs pneumatiques contrôlés pour produire une force de poussée.	Un réseau de microactionneurs pneumatiques contrôlés pour produire une force magnétique
<i>Vélocité</i>	Lente	Moyenne	Bonne	Assez bonne
<i>Précision</i>	Bonne	Assez bonne	Contrôle peu précis	Contrôle peu précis
<i>Capacité de charge</i>	Faible	Bonne	Assez moyenne	Faible
<i>Miniaturisation</i>	Excellente	Très bonne	Moyenne	Plutôt faible
<i>Coût</i>	Moyen à élevé	Correct	Élevé	Élevé à très élevé pour de forte capacité

Tab. 2.1 – Comparaison entre micromanipulateurs 2-D « à contact » et « sans contact »

essentiel à l'évolution de ces technologies de micromanipulateurs 2-D vers des applications industrielles. Il s'agira donc de réaliser des dispositifs de « *smart surface* » sur la base de « *smart structure* » comme définie dans la Chapitre 1. Dans ce domaine, et à notre connaissance, il n'existe qu'une seule réalisation de « *smart surface* » développée à l'échelle micrométrique.

Dès 1999, poursuivant les premiers travaux de *Böhringer* sur des modèles de micromanipulation distribués, *J. W. Suh* et ses collègues chercheurs de l'Université Stanford et de Washington (États-Unis) vont mettre au point le premier prototype intégrant capteurs, contrôleurs et microactionneurs sur un même circuit CMOS [49]. L'architecture globale de ce dispositif comprenait quatre puces permettant de combiner 256 cellules de forces distribuées sur une surface de $16 \times 16 \text{ mm}^2$, comme le montre la figure 2.5. Les microactionneurs employés sont de types ciliaires bimorphes thermiques à quatre directions.

Cette réalisation d'intégration reste à l'heure actuelle le prototype de « *smart surface* » le plus accompli. Elle a permis de convoier avec une directivité satisfaisante des objets en silicium ayant une épaisseur de $100 \mu\text{m}$ pour des niveaux d'alimentation électrique de $3,5 \text{ V}$ à 6 V . Des difficultés de convoiage sont tout de même apparues pour des charges gravitationnelles unitaires excédant 10 N/mm^2 . La vitesse de convoiage était de l'ordre de 1 mm/s .

Grâce à ce premier prototype intégré, l'idée de la « *smart surface* » était née. Cette idée se fondait sur le postulat suivant ; l'interconnexion parallèle de nombreux modules intégrés possédant leurs propres circuits logiques internes permet de réduire le volume d'information échangé avec l'extérieur. Une telle architecture aboutit à une simplification sensible de l'infrastructure électromécanique du dispositif, comparativement à une solution classique, centrée sur le contrôle

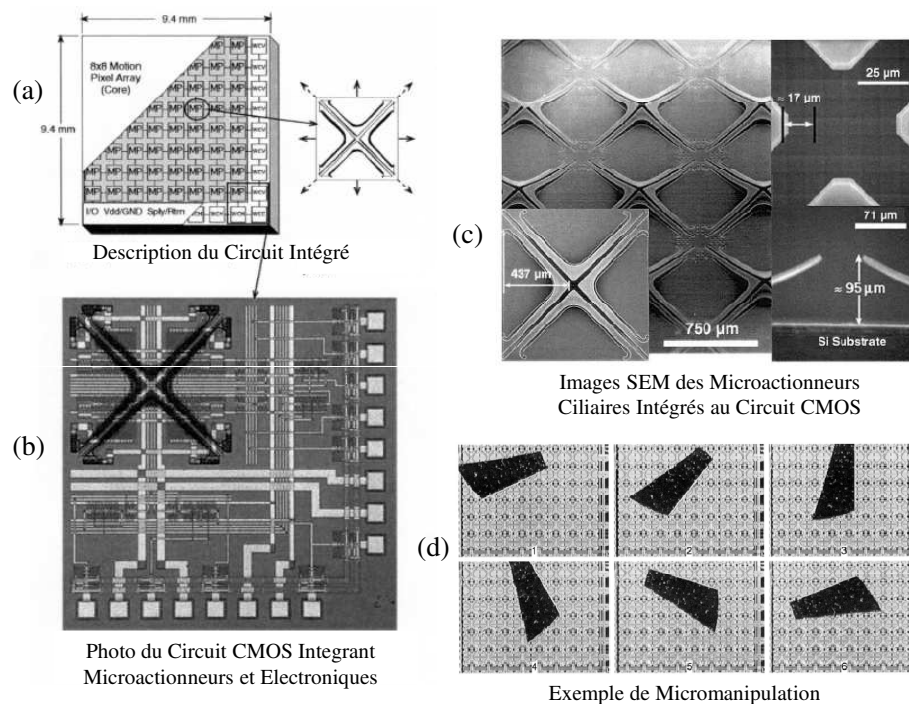


Fig. 2.5 – *Prototype de « smart surface » ou micromanipulateur 2-D intégrant capteurs, contrôleurs et microactionneurs sur un même circuit CMOS (images extraites de l'article [49])*

individuel de chaque actionneur depuis une unité de pilotage décentralisée, comme le montre la figure 2.6.

En théorie, chaque module possède l'intelligence suffisante pour contrôler son propre actionneur et pour coopérer avec les modules voisins. Même si la sphère d'influence d'un actionneur demeure intrinsèquement limitée, la coopération de nombreux modules aboutit à l'exécution de tâches éventuelle complexes telles que, par exemple, le positionnement parallèle d'objets multiples initialement décrit.

2.1.4 Tendances

La synthèse que nous avons tabulée précédemment rend bien compte de l'état de l'art au début des années 2000 du domaine des « surfaces actives » à base de MEMS. Depuis, si aucune révolution notoire n'est venue bouleverser les performances de convoyage enregistrées par les dispositifs destinés au domaine du micro- et nano-usinage, nous avons cependant assisté à une amélioration progressive de ces dispositifs.

Par exemple, l'utilisation de wafer type SOI (Silicon on Insulator) a permis de franchir une étape importante dans la solidité et la fiabilité des dispositifs de micromanipulation 2-D [37]. De même, l'emploi de plus en plus fréquent de polymères commerciaux (SU-8, PMMA, Parylène, etc.) a été à l'origine de composants à bas coût de fabrication, comme pour la technologie

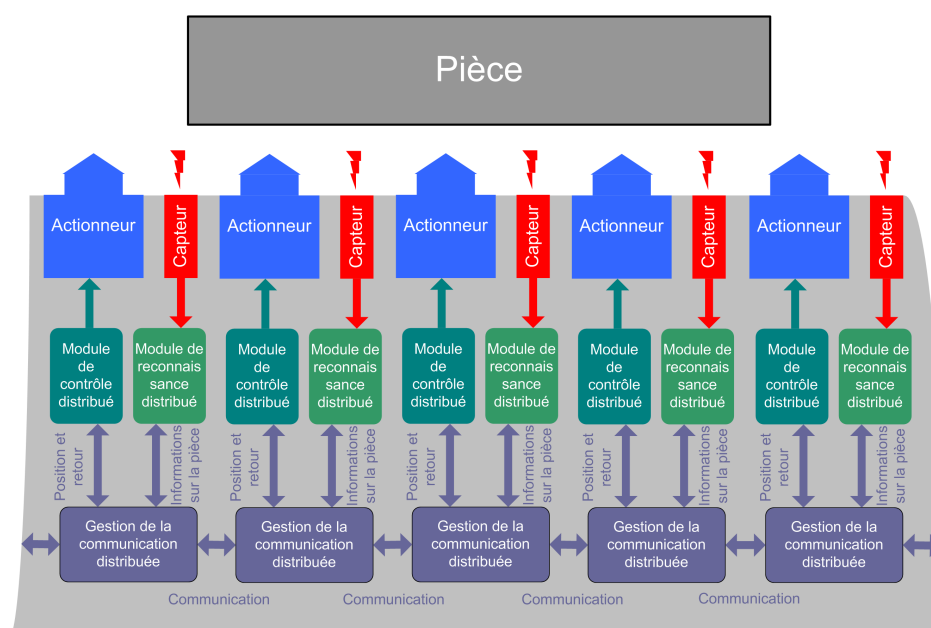


Fig. 2.6 – *Concept de « smart surface »*

de système « sans contact » de type pneumatique [50]. On assiste également à l'utilisation de technologies de plus en plus inspiré par la « *Nature* » pour de nouvelles applications où la manipulation se réalise en milieu contraignant comme les milieux liquides [51].

Dans le domaine des « smart surface », bien que de nombreux progrès aient été enregistrés en technologie de fabrication CMOS et « post-process » CMOS associé aux MEMS, aucune réalisation concrète d'intégration n'a été enregistrée à notre connaissance. La tendance serait plutôt à l'intégration de structures hybrides 3-D, ce qui pourrait être la solution d'avenir face à la difficulté et au coût des structures monolithiques [II.17], [II.18]. De plus, l'assemblage 3-D semble répondre à un besoin de confinement inhérent au concept même des « surfaces actives » (plus de surface utile).

Les approches de conception de « smart surface » de façon monolithique 2-D ou hybrides 3-D sont des thèmes de recherche de plus en plus « tendances » aujourd'hui. En effet, les récents développements dans le domaine des « surfaces actives » à base de MEMS distribués se sont orientés vers de nouvelles applications comme la micro- et nano-manipulation d'objets biologiques (cellule, ADN, etc.) [52]. Ces récentes tendances impliquent de nouvelles technologies comme les matériaux souples et conducteurs (polymères conducteurs), les phénomènes physico-chimique comme la diélectrophorèse, l'électrowetting, l'électrowetting-on-dielectric (EWOD), etc. [53].

Ces domaines de recherche sont en pleine expansion et l'apport des technologie de « smart surface » pourrait être essentiel à leur émancipation.

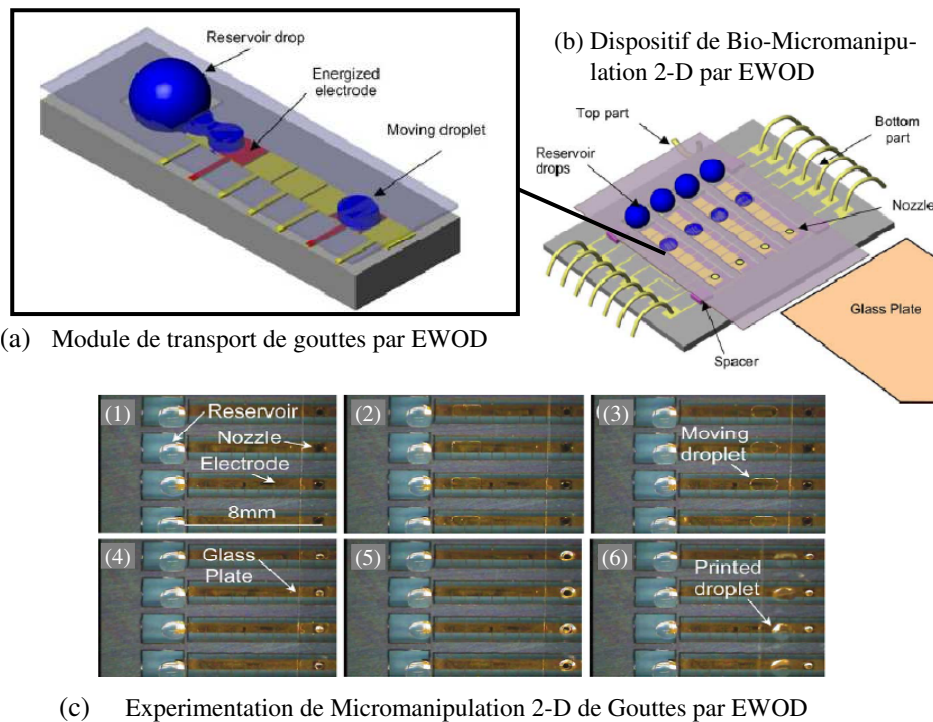


Fig. 2.7 – Dispositif de bio-micromanipulation 2-D par EWOD (images extraites de l'article [54])

2.2 Prototype à champ de forces fluidiques

2.2.1 Introduction

Une application innovante des microsystèmes distribués intelligents et autonomes est proposée dans ce chapitre à travers la conception à base de technologies MEMS d'une « *surface active* » dédiée à la micromanipulation 2-D « sans contact » et pneumatique. Pour cela, une matrice de microactionneurs fabriquée sur silicium a été développée pour la réalisation d'un système de micro-convoyage par écoulement d'air. Nous parlerons plus communément de réseaux de microactionneurs pneumatiques pour application de micromanipulation parallèle massive. Ce type de système pourrait être imaginé dans un secteur-clé du développement futur des microsystèmes, soit la manipulation de microcomposants industriels à une haute échelle de productivité.

2.2.1.1 Contexte scientifique

Nos recherches se sont basées sur les travaux réalisés au sein du laboratoire du *Pr. Fujita* (Fujita Lab.) de l'Institute of Industrial Science (IIS) de l'Université de Tokyo au Japon. En effet, ce dernier explore depuis plus d'une dizaine d'années les aspects de microfabrication et de contrôle de réseaux de MEMS pour « surfaces actives ». Plus particulièrement, son groupe a récemment achevé l'élaboration d'une technologie pour la conception de réseaux de microac-

tionneurs pneumatiques à base de technologie silicium [37].

Le rapprochement avec le laboratoire du *Pr. Fujita* a pu se faire à travers la délégation, en qualité de chargé de recherche CNRS, de *Y.-A Chapuis* au sein du LIMMS (Laboratory of Integrated Micro Mechanical Systems) qui est une Unité Mixte Internationale de recherche du CNRS (CNRS UMI 2820) associé avec l'Institute of Industrial Science (IIS) de l'Université de Tokyo au Japon.

La fonction d'une « micromanipulation distribuée » « sans contact », de type fluïdique, est de développer des flux (ou jets) d'air, à l'aide de microactionneurs pneumatiques, pour générer des forces directionnelles en vue de convoyer des objets avec une friction parasite minimale ou en lévitation.

Le principe et la technologie d'un tel système diffère donc fondamentalement d'un dispositif type « à contact », puisque ce dernier utilise la friction pour manipuler un objet sur un plan de manière coopérative interactive. Dans notre cas de manipulation fluïdique, la direction du flux d'air est contrôlée par l'ouverture et la fermeture de deux canalisations (ou orifices) d'air à travers l'actionnement de micro-valves, comme le montre la figure 2.8.

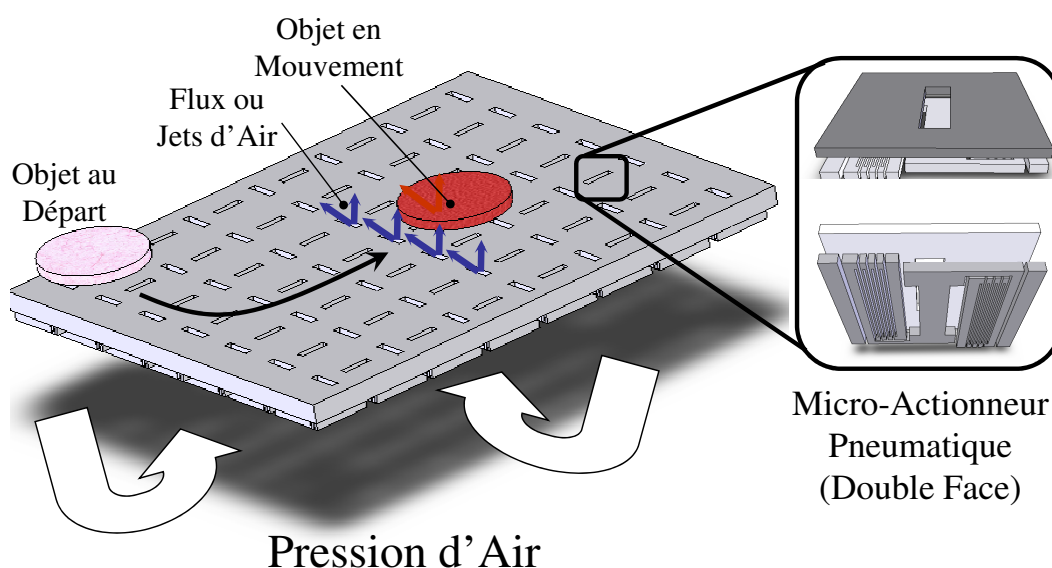


Fig. 2.8 – Dispositif de réseaux de microactionneurs pneumatiques en technologie MEMS pour micromanipulation 2-D.

2.2.2 Approche de conception

2.2.2.1 Principes de fonctionnement

Les principes de fonctionnement du microactionneur pneumatique sont présentés sur la figure 2.9. L'originalité de cette structure réside dans l'approche à deux couches proposées. On observe ainsi que le microactionneur pneumatique se compose :

- D'une couche fonctionnelle supérieure composée d'un simple orifice, comme le montre la figure 2.9(b),
- D'une couche fonctionnelle inférieure composée d'une micro-valve mobile actionnée par un microactionneur électrostatique, dimensionnée et alignée sur la forme de l'orifice de la couche supérieure, comme le montre la figure 2.9(c).

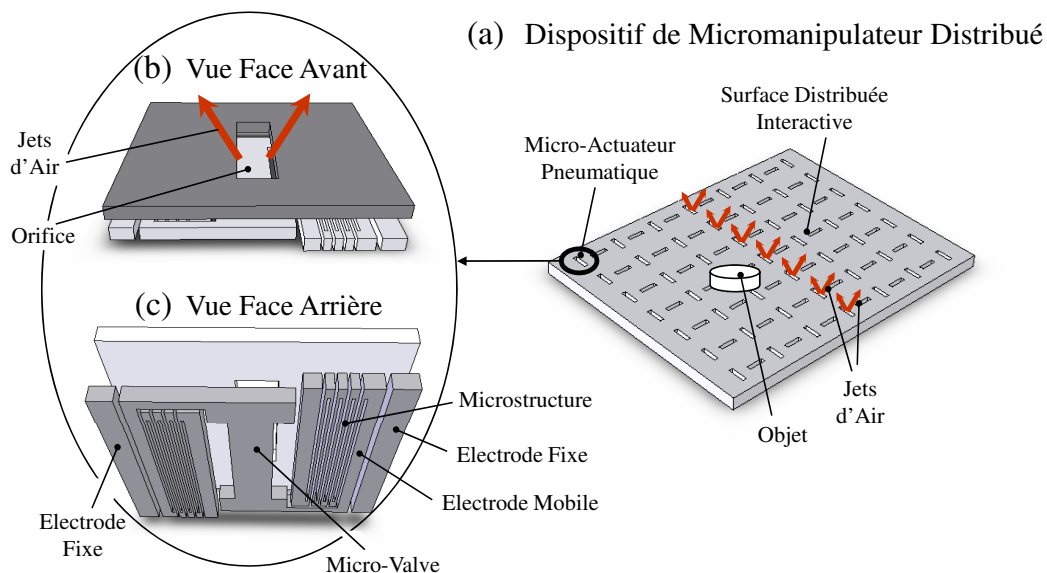


Fig. 2.9 – Dispositif de réseaux de microactionneurs pneumatiques pour micromanipulation 2-D à partir de technologie SOI

Comme nous pouvons le constater, cette technologie nécessite une approche de fabrication à deux couches. Nous emploierons pour cela des *wafers* SOI qui se composent de deux substrats en silicium (haut et bas) d'une même épaisseur. Dans le substrat supérieur (côté haut), un trou a été conçu, tandis que le substrat inférieur (côté bas) inclut la structure de la micro-valve mobile basée sur le microactionneur électrostatique.

Les deux électrodes qui ont pour objectif de fournir la force électrostatique sont alignées à côté des parties mobiles et fixées au substrat supérieur. La partie mobile se compose d'une électrode mobile, qui est en face de l'électrode fixe pour la tension électrique, et des poutres suspendues du microactionneur. Les poutres suspendues sont reliées à l'électrode de masse par la prolongation et l'alignement de celui-ci. L'électrode de masse est fixée au substrat supérieur, et sa partie prolongée sera également utilisée comme un stoppeur. Ainsi, la partie mobile est arrêtée avant qu'elle n'entre en contact avec l'électrode, des problèmes de court-circuit sont donc évités.

La figure 2.10 montre la vue transversale et la vue de haut de la structure élémentaire du microactionneur pneumatique dans deux cas de fonctionnement. Quand il n'y a aucune tension électrique, un léger flot d'air vertical est produit pour maintenir la lévitation de l'objet, comme l'illustre la figure 2.10(a). En revanche, lorsqu'une tension électrique est appliquée sur l'électrode fixe de droite du microactionneur électrostatique, la micro-valve s'ouvre en partie sur la droite,

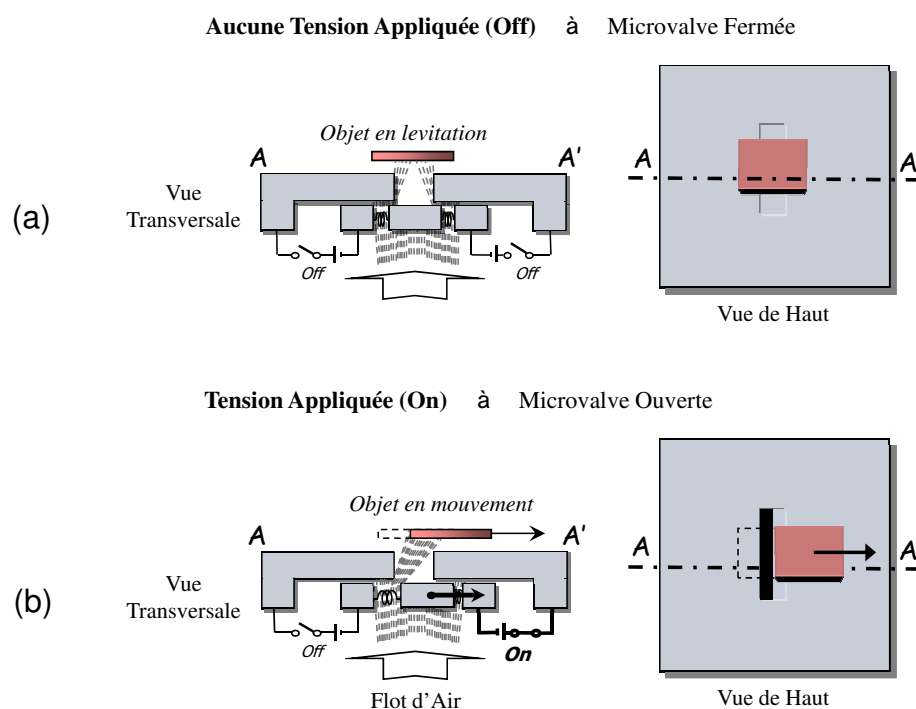


Fig. 2.10 – *Principes de fonctionnement et de conception du microactionneur pneumatique*

dégageant l'orifice supérieur et permettant la génération d'un jet d'air orienté vers la droite, comme le montre la figure 2.10(b).

2.2.2.2 Modèle à base de calculs par éléments finis

Pour valider et caractériser la fonctionnalité de notre concept de microactionneur, un modèle à base de calculs par éléments finis a été élaboré sous le logiciel commercial ANSYS® [37]. La figure 2.11 présente les résultats de simulation du microactionneur pneumatique modélisé dans deux zones de pression distinctes avec les dimensions de l'orifice ($200 \mu m$ de large) et de la micro-valve (de 5 à $20 \mu m$) du dispositif.

Ces résultats montrent l'écoulement et la vitesse du flot d'air dans les deux cas de fonctionnement de la micro-valve ; au repos, comme le montre la figure 2.11(a), actionnée (à droite), comme le montre la figure 2.11(b).

Les figures 2.11(c) et (d) donnent l'évolution de la vitesse verticale et latérale du flot d'air obtenue en sortie du microactionneur pneumatique pour une pression d'air en entrée de $600 Pa$.

On notera que la direction et l'amplitude de la vitesse du flot d'air est très variable selon la position où elle est calculée. Il est donc très difficile de donner une valeur exacte de vitesse. Par contre, considérant la latence de la micro-valve (temps d'application de la force fluïdique), on pourra considérer que la valeur maximale de la vitesse latérale est en moyenne toujours appliquée sur la tranche de l'objet en déplacement au-dessus du microactionneur. Par exemple, dans le cas de la figure 2.11(d), le flot d'air qui s'applique sur l'objet est approximativement de

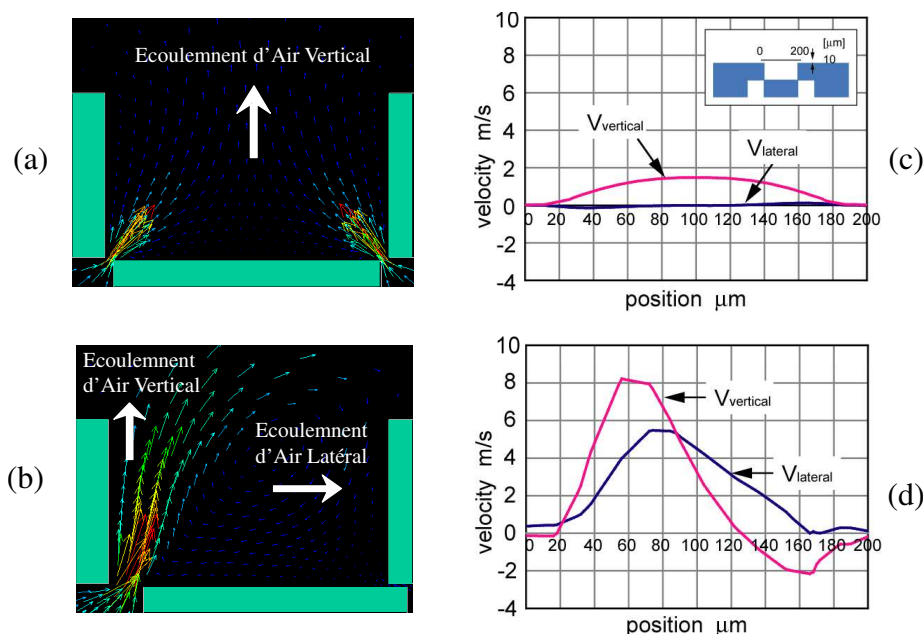


Fig. 2.11 – Simulation FEM du fonctionnement fluidique du microactionneur pneumatique.

5,5 m/s au point maximal de vélocité latérale.

2.2.2.3 Processus de fabrication

La figure 2.12 montre les principales étapes de microfabrication sur silicium du microactionneur pneumatique. Le substrat de base employé ici est un *wafer* type SOI d'épaisseurs 100/2/100 μm , comme illustré dans la figure 2.12(a).

Dans la seconde étape du processus de microfabrication, l'ouverture du composant (orifice) est formée, après lithographie et dépôt d'un masque photosensible, par une gravure sèche par plasma ou DRIE (*Deep Reactive Ion Etching*), comme l'illustre la figure 2.12(b).

Utilisant les mêmes techniques, les motifs du microactionneur électrostatique de la couche inférieure, sont réalisés en alignant la micro-valve avec l'orifice conçu précédemment, comme illustré sur la figure 2.12(c).

Finalement, comme présenté dans la figure 2.12(d), les microstructures mobiles sont libérées par gravure en surface de la couche intermédiaire du *wafer SOI*. Pour cette étape de gravure, la technique à base de vapeur d'acide fluorhydrique (Vapor HF) est appliquée pour éviter les problèmes de « stiction ».

Pour plus de détails du processus de fabrication, il est recommandé de se référer à l'article [37].

La figure 2.13 présente les résultats de microfabrication des faces avant et arrière du dispositif par les images obtenues par microscope électronique de balayage (SEM). La figure 2.13(a) montre une image du chip MEMS final et de son boîtier de connections électronique pour le pilotage

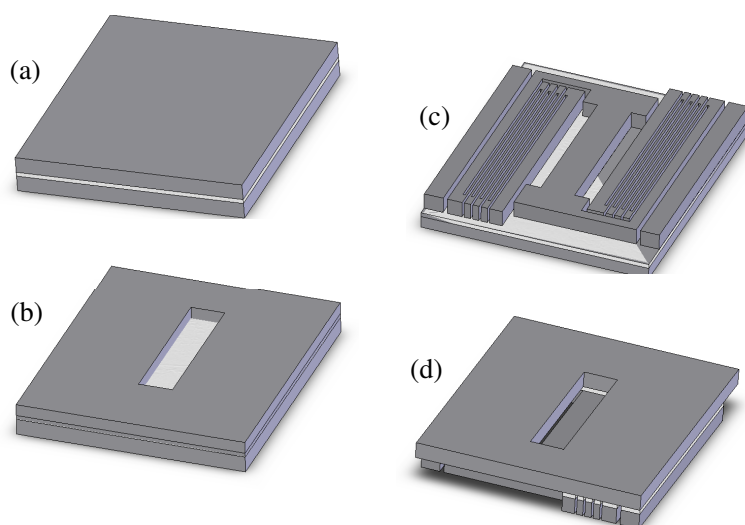


Fig. 2.12 – *Processus de fabrication du microactionneur pneumatique.*

des tensions d'alimentation des micro-actionneurs.

La taille du dispositif est d'environ $35 \times 35 \text{ mm}^2$ pour 560 microactuateurs pneumatiques en technologie MEMS. Les images SEM de la figure 2.13(b) montrent les résultats de microfabrication de la couche active fluide et l'arrangement des réseaux de micro-actionneurs. L'image SEM de la figure 2.13(c) montre la face arrière de la matrice de microactuateurs pneumatiques. La taille de chaque cellule actionneur est bien approximativement de $1 \times 1 \text{ mm}^2$.

2.2.3 Approche de contrôle

2.2.3.1 Structure générale

La figure 2.14 illustre la structure générale de contrôle du système distribué autonome de la matrice de microactionneurs pneumatiques. Elle se compose d'un système vidéo et d'une unité centrale de calcul type PCU d'où sera traité l'ensemble des données de perception et de localisation en vue d'allouer individuellement les ordres de contrôle aux microactionneurs pneumatiques. Une interface de pilotage (driver) servira au multiplexage et à l'alimentation $5/120 \text{ V}$ de chaque microactionneur.

Le microsystème contenant 560 microactionneurs n'est pas encore complètement décentralisé. Donc, pour la première approche, nous proposons de commander les microactionneurs rangées par rangées, colonnes par colonnes, ce qui limite les connexions à 128, où nous pouvons appliquer la stratégie de la prise de décision décentralisée. Ainsi, les signaux de contrôle à 8 bits émis par le PCU peuvent être envoyés vers le microsystème, par l'intermédiaire d'un PIC lequel pilote plusieurs « HV drivers », qui consiste en un 4 32-bit parallèle carte de sortie avec 120 V-driver ICs . Tous les *drivers ICs* (Toshiba, TD62981P) sont intégrés dans un prototype de carte d'interface interne unique et leur fournissent une tension d'alimentation fixée au maximum

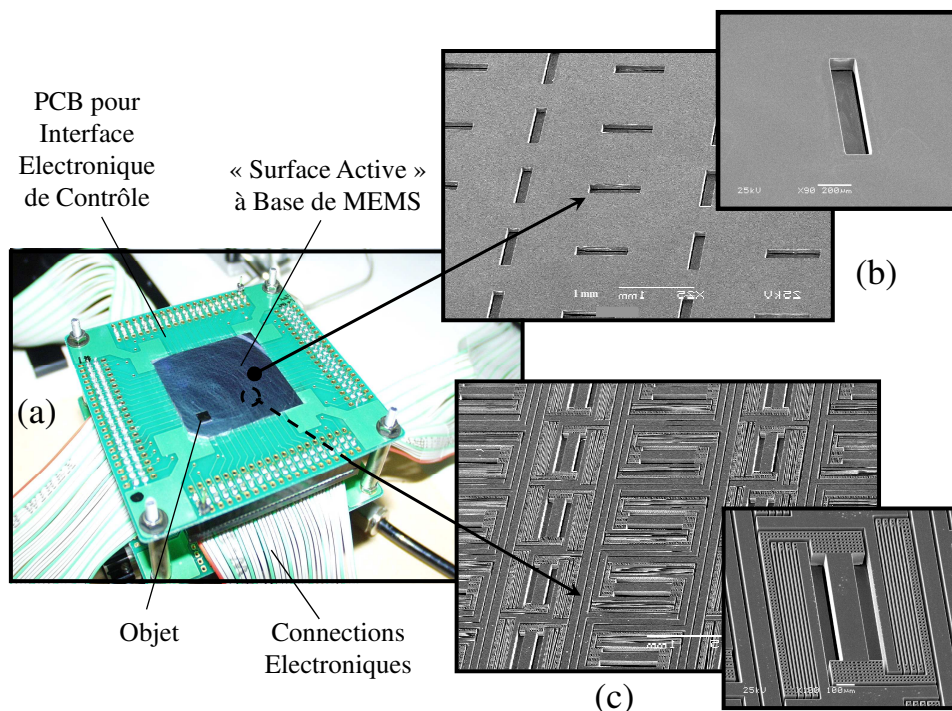


Fig. 2.13 – Images de la matrice de microactionneurs pneumatiques formant la « surface active ». (a) Dispositif avec son support de connexions électroniques. (b) Images SEM de la face-avant de la « surface active ». (c) Images SEM de la face-arrière de la « surface active ».

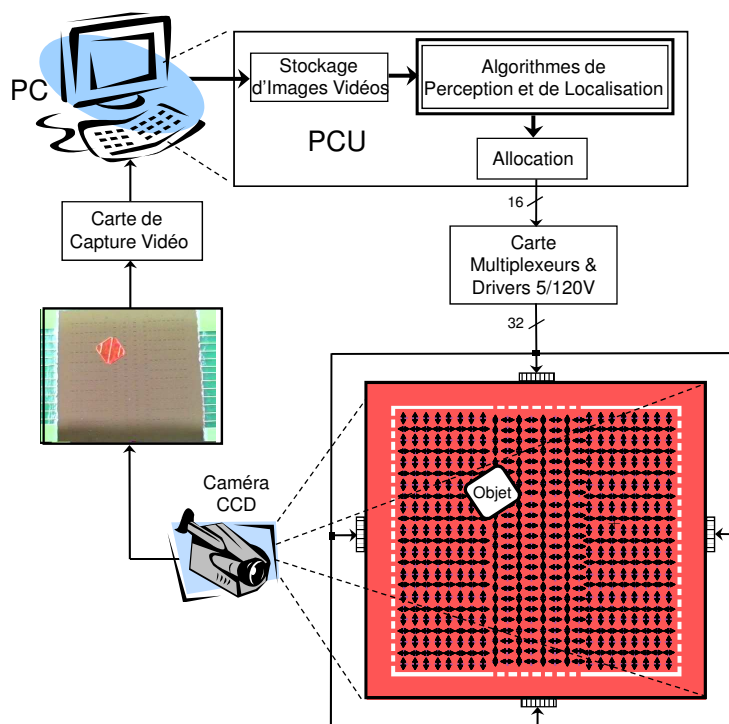


Fig. 2.14 – Structure de contrôle de la matrice microactionneurs pneumatiques.

120 V DC.

2.2.3.2 Flot de contrôle en boucle fermée

Le système est supposé être autonome. Ce qui signifie que chaque microactionneur devrait pouvoir détecter si un objet le couvre ou pas. Pour rendre cette hypothèse possible, nous proposons de générer une image codée spécifique de la surface distribuée où chaque pixel peut être affecté à un microactionneur du système. Ce sera une image à deux dimensions dont chaque élément est codé soit par « 1 », soit par « 0 ». Concrètement, quand un microactionneur pneumatique est couvert par un objet, elle est codée par « 1 », et dans le cas contraire, elle est codée par « 0 ».

Pour ce faire, une caméra CCD avec une résolution de 640×480 (Vainqueur-JVC, GR-DVP7) capture des séquences vidéo de la surface active distribuée avec l'objet en mouvement par l'intermédiaire d'une carte vidéo (I-O DATA, GV-BCTV5/PCI). Toutes les séquences vidéo sont enregistrées dans un fichier en format *Device Independent Bitmap* (DIB), puis vont être traitées par un algorithme de traitement écrit en langage C.

Cette méthode de commande est reprise dans le flot de conception de la figure 2.15. On distingue ainsi quatre étapes principales dans le processus de micromanipulation de la matrice en boucle fermée :

- **La perception** : soit le traitement par processeur de l'image numérique élaborée à partir de la caméra CCD pour la reconnaissance de l'objet.
- **La localisation** : soit l'utilisation des techniques de représentation par quadrillage-plan correspondant au réseau de microactionneurs pneumatiques, aussi appelé : « grid map représentation ». La représentation à trois niveaux de la figure 2.15 décrit schématiquement cette approche.
- **La stratégie de contrôle** : soit l'élaboration d'un algorithme de contrôle des microactionneurs, selon la destination de l'objet et sa position par rapport aux microactionneurs pneumatiques.
- **La modulation de contrôle** : soit la modulation de contrôle des microactionneurs pour permettre le bon déplacement mécanique de l'objet.

2.2.3.3 Stratégie de contrôle décentralisé

a. Modèle décentralisé

L'application d'une stratégie de contrôle décentralisé va nous permettre d'élaborer une méthode de détection de l'extrémité de l'objet en mouvement. Cette technique est largement employée dans le domaine du traitement vidéo lorsque l'on veut détecter un mouvement à partir d'images CDD. Dans le cas de la « surface active » étudiée, cette technique a été appliquée de manière simplifiée [37]. Nous allons en donner les principes de contrôle dans ce qui suit.

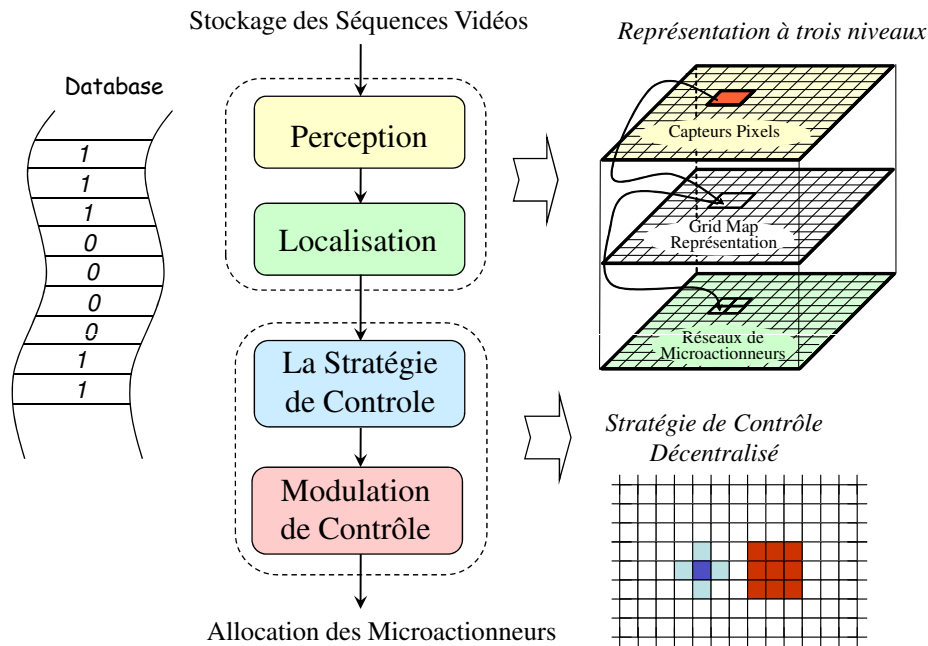


Fig. 2.15 – Flot de contrôle en boucle fermée du réseau de microactionneurs

Le bord de l'objet est la partie la mieux exposée à la force de poussée lors de la génération d'un jet d'air par l'activation du microactionneur pneumatique. En effet, à partir d'une étude de l'écoulement de l'air sur l'objet, nous savons que le coefficient aérodynamique le plus élevé qui s'applique sur l'objet correspond à la tranche des bords de ce dernier [55]. Il s'agit donc de détecter l'extrémité de l'objet, lorsque celui-ci a couvert le détecteur avant de le dépasser, et d'appliquer immédiatement un jet d'air pour avoir la meilleure réponse de poussée.

La stratégie de contrôle décentralisé pour détection de bord se base sur un modèle de « surface active » représentée par des rangées (de $i = 1$ à $i = i_{max}$,) et des colonnes (de $j = 1$ à $j = j_{max}$,) de cellules. On désignera chaque cellule avec la notation $C_{i,j}$. Chacune des cellules doit respecter les cinq règles de contrôle suivantes :

1. Chaque cellule $C_{i,j}$ est supposée intelligente pour prendre une décision et capable de mémoriser certaines informations,
2. Chaque cellule $C_{i,j}$ est supposée sensitive et capable de détecter un objet qui la couvre,
3. Chaque cellule $C_{i,j}$ est capable de déplacer un objet dans deux directions en actionnant la partie microactionneur qui la concerne,
4. Chaque cellule $C_{i,j}$ est supposée avoir le même voisinage, excepte pour les cellules des bords du dispositif,
5. Chaque cellule $C_{i,j}$ est capable de communiquer avec ses 4 cellules voisines ($C_{i-1,j}$, $C_{i,j+1}$, $C_{i,j-1}$, $C_{i+1,j}$).

La figure 2.16 illustre la représentation des cellules $C_{i,j}$ du modèle de la « surface active ».

b. Stratégie de contrôle

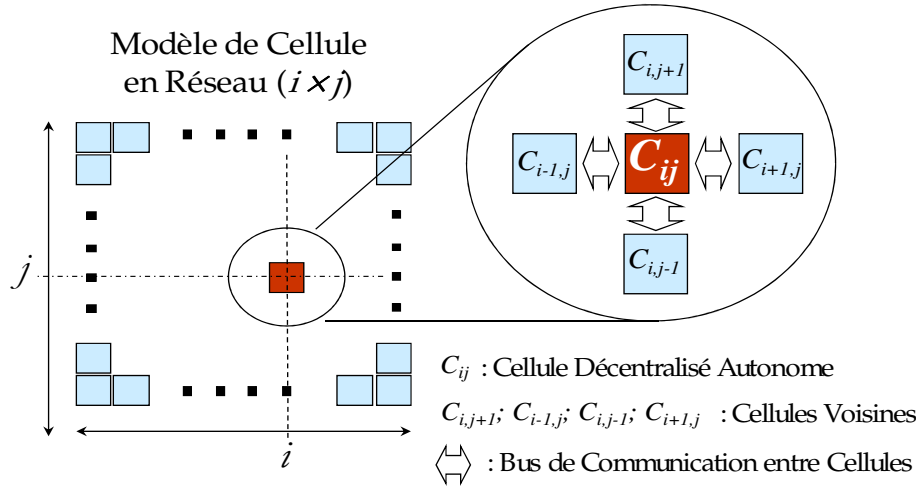


Fig. 2.16 – Représentation des cellules $C_{i,j}$ du modèle de la « surface active »

La même stratégie de contrôle décentralisé a été implantée dans chaque cellule $C_{i,j}$ de manière à leur permettre de détecter le passage du bord arrière de l'objet et de prendre une décision d'actionnement. Cette stratégie est illustrée par la séquence de détection décrite sur la figure 2.17.

Pour comprendre le raisonnement suivi, nous avons choisi de présenter deux séquences vidéos du passage d'un objet au-dessus d'une « surface active » représentée par un réseau de microactionneurs pneumatiques : (1) La séquence de référence ($n - 1$); (2) La séquence courante (n). Entre les deux séquences, la détection du bord de l'objet va être opérée par le groupe de cellules se trouvant à l'endroit décrivant le bord de l'objet. On emploiera un système de codage pour distinguer les cellules couvertes par l'objet et celles non-couvertes :

- Cellule couverte par un objet : $C_{i,j} = \langle 1 \rangle$
- Cellule non- couverte par un objet (« surface active ») : $C_{i,j} = \langle 0 \rangle$

Ainsi, lorsqu'à la *séquence courante* (n), une cellule ($C_{i,j}$) est codée et testée comme supérieure (ex : « 1 ») à l'une de ses quatre cellules voisines ($C_{i-1,j}$, $C_{i,j+1}$, $C_{i,j-1}$, $C_{i+1,j}$) (ex : « 0 »), cela signifie qu'un bord d'objet a été détecté, comme le montre la figure 2.17(2).

Pour savoir quel bord de l'objet a été détecté, il s'agit, par exemple lorsque l'objet se déplace de la gauche vers la droite, de connaître « l'état » ou le code de la cellule à la séquence précédente, soit la séquence de référence ($n-1$), comme le montre la figure 2.17(1).

On voit bien dans notre cas de détection que la cellule ayant détecté un bord de l'objet à la *séquence courante* (n) est codée « 1 », de la même manière qu'à la *séquence de référence* ($n-1$). Cela signifie que le bord de l'objet détecté est le bord sortant de la zone de détection. Si le code avait été « 0 » à la séquence de référence ($n-1$), nous aurions détecté le bord entrant de l'objet. Chacune des cellules ayant détecté le bord de l'objet va alors activer la commande de la micro-valve du microactionneur pneumatique qui lui est allouée, comme le montre la figure 2.17(3). En conséquence, la rangée de jets d'air générés va produire le champ de forces fluidiques

permettant à l'objet de se déplacer dans la direction désirée.

2.2.4 Validation expérimentale : Caractérisation

2.2.4.1 Mise en oeuvre du banc fluide expérimental

La figure 2.18 représente la structure expérimentale du banc d'essai utilisée pour la validation du système.

L'installation expérimentale pour le micromanipulateur pneumatique distribué se compose du système de contrôle de l'air comprimé, de l'alimentation à haute tension et du traitement par ordinateur. Le dispositif microactionneur est posé sur une plateforme mécanique pour ajuster la position d'équilibre de la surface pneumatique. La source du flux d'air est fournie par le gaz comprimé d'azote (N_2) à travers des valves de contrôle. A partir de cela, avec un système d'interrupteur adapté et un générateur de fonction, nous pouvons effectuer une impulsion périodique de flux d'air à 5 Hz de "Marche/Arrêt" afin de réduire l'instabilité de lévitation due à la dynamique fluide du système d'atténuation.

Les détails du système de flux d'air comprimé sont montrés dans la figure 2.18. La plage de la pression du flux d'air est approximativement de 3 kPa à 14 kPa . En effet, une limite inférieure est nécessaire pour assurer la lévitation (3 kPa), juste comme une limite supérieure est fixée pour éviter au dispositif d'être endommagé.

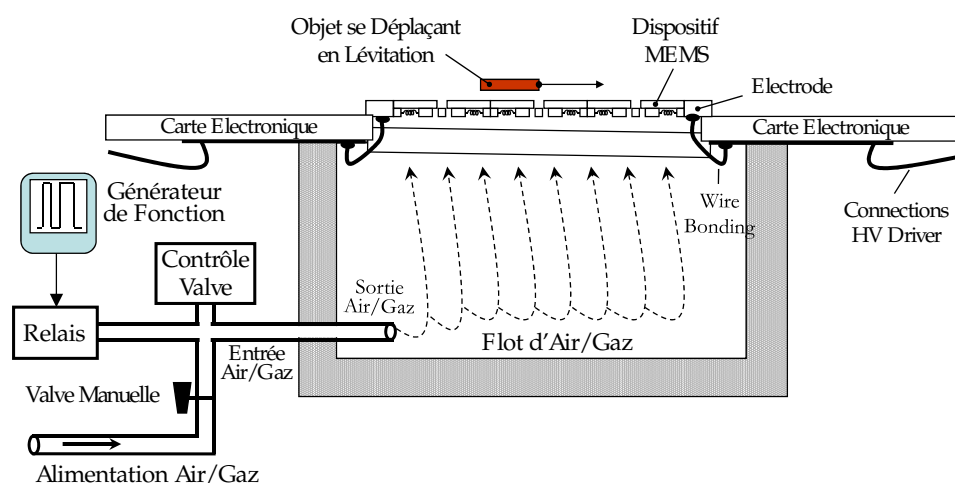


Fig. 2.18 – Illustration du banc fluide expérimental

2.2.4.2 Résultats de convoyage 2-D en boucle ouverte

La figure 2.19 donne les résultats expérimentaux obtenus lors du convoyage en boucle ouverte d'un objet plat en silicium.

Les dimensions 3-D de l'objet en question était de $2,1 \times 4,1 \times 0,2\text{ mm}^3$ pour un poids

approximatif de 2 mg . L'alimentation fluide est opérée par un gaz d'azote (N_2) amené à la pression de $13,2 \text{ kPa}$. Les microactionneurs électrostatiques des micro-valves sont pilotés par des tensions de 90 V .

Nous avons prédéfini le chemin de convoyance de l'objet en activant les microactionneurs adéquats. Le but est donc de convoier l'objet de sa position d'origine jusqu'à la position désirée qui se trouve au centre de la « surface active ». On obtient alors un convoyage 2-D de l'objet avec une vitesse estimée à 10 mm/s .

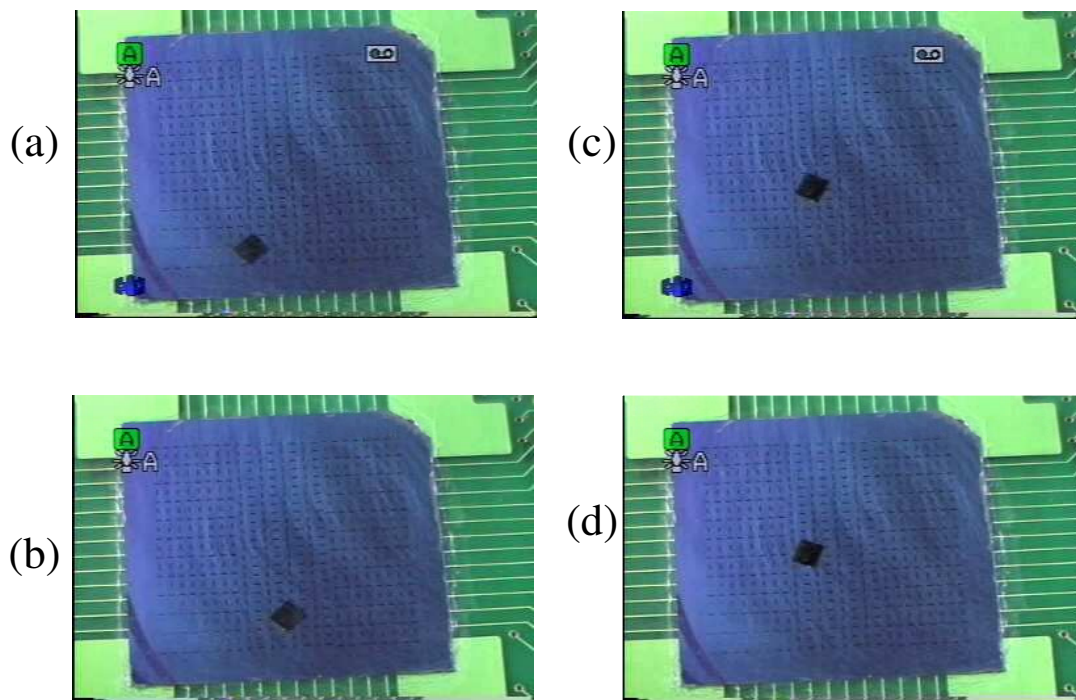


Fig. 2.19 – Résultats expérimentaux d'une phase de convoyage 2-D en boucle ouverte

2.2.4.3 Résultats de convoyage 2-D en boucle fermée

Les résultats expérimentaux en boucle fermée du convoyage d'un objet plat en plastique sont présentés sur les 5 séquences vidéo de la figure 2.20.

Ces résultats ont été obtenus en appliquant la stratégie de contrôle décentralisé décrite dans ce chapitre (Cf. 2.2.3.3). Les dimensions de l'objet manipulé sont les suivantes : $5,6 \times 5,6 \times 0,25 \text{ mm}^3$ pour un poids de $11,4 \text{ mg}$. La pression et la nature fluide appliquée sont données par un gaz d'azote (N_2) alimenté à la pression de $13,2 \text{ kPa}$. Pour des raisons de stabilité de l'objet, on opérera un « damping » du gaz injecté, soit une modulation de pression en « marche/arrêt » à la fréquence de 5 Hz . La tension appliquée est de 90 V pour le pilotage des micro-valves par effet électrostatique.

A travers les 5 séquences vidéo de la figure 2.20, on observe, entre deux rangées de cellules, les phases de détection du bord de l'objet et la génération des forces fluidiques de poussée après

l'application des ordres de commande des microactionneurs pneumatiques.

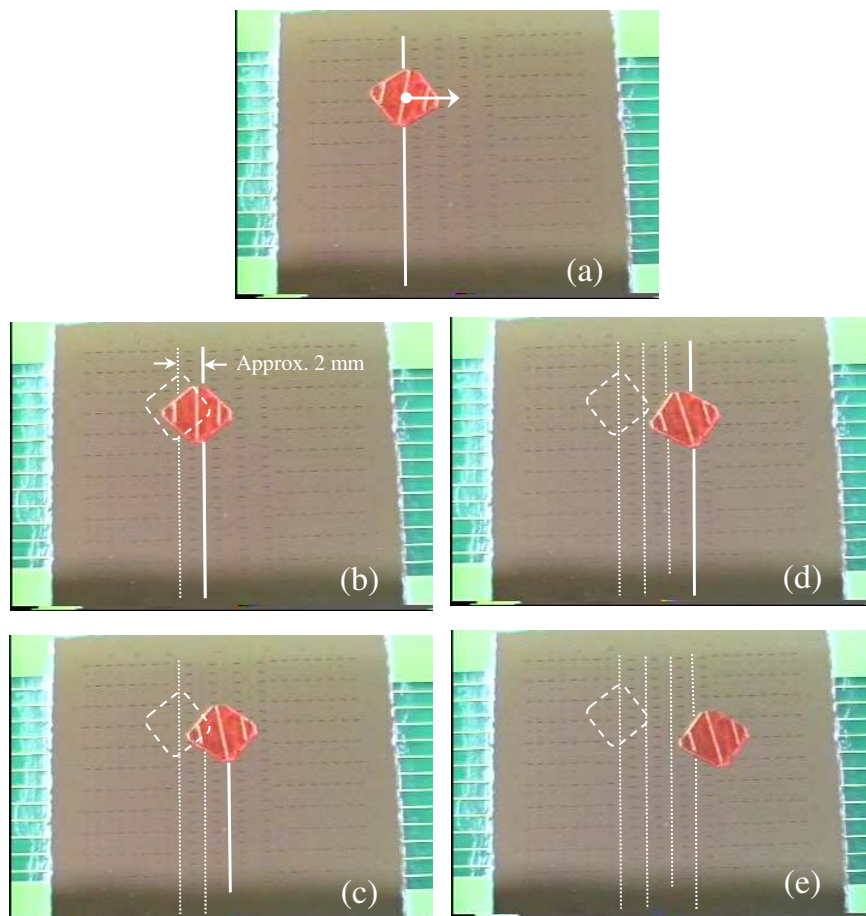


Fig. 2.20 – Résultats expérimentaux d'une phase de convoyage en boucle fermée

Des tests de caractérisation des performances de micromanipulation 2-D du dispositif étudié ont été réalisés à partir de deux objets plastiques de dimension et poids différents; objet A ($5,6 \times 5,6 \times 0,25 \text{ mm}^3$, $11,4 \text{ mg}$), objet B ($4,5 \times 4,1 \times 0,25 \text{ mm}^3$, $6,6 \text{ mg}$). On montre que la vitesse maximale de l'objet peut atteindre $4,5 \text{ mm/s}$ pour une pression de gaz de $13,2 \text{ kPa}$. De même, la capacité de charge de la « surface active » a été mesurée à $0,37 \text{ }\mu\text{N}$ maximum.

2.3 Conclusion

A travers l'état de l'art et la présentation du prototype étudié dans ce chapitre, le choix des micromanipulateurs 2-D et « smart surface » s'avère parfaitement en adéquation avec les objectifs de recherche poursuivis dans cette thèse. En effet, les contraintes multi-technologiques et multi-physiques de tels dispositifs et de leur environnement sont très bien adaptées au défi de modélisation et simulation multi-domaines que nous souhaitons relever.

La technologie des micromanipulateurs 2-D et « smart surface » à base de MEMS distri-

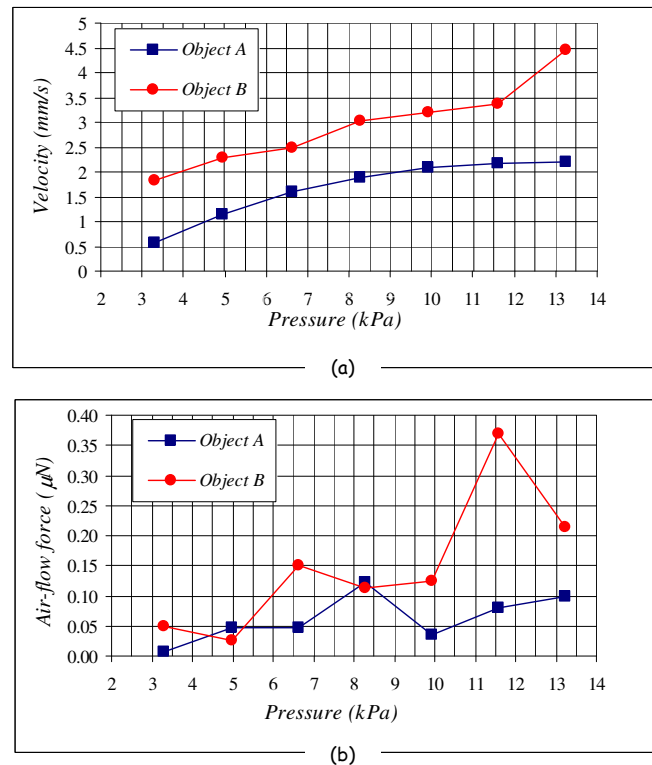


Fig. 2.21 – Résultats de performance vitesse (a) et de force de poussée fluidique (b) du dispositif de « surface active »

bués est un domaine en pleine expansion avec des possibilités d'application dans de nombreux autres secteurs tels que la biologie et le médical.

Ce chapitre a aussi permis d'étudier, en détail, le prototype de micromanipulation 2-D à base de champs à forces fluidiques qui a été développé au sein du laboratoire du *Pr. Fujita* de l'Université de Tokyo au Japon lors de la période de détachement CNRS au LIMMS de *Y.-A. Chapuis* (co-encadrant de ce travail de recherche).

Plusieurs résultats expérimentaux du prototype de micromanipulation ont ainsi pu être mis en évidence et analysés, de même que des résultats de simulation à base de calculs par éléments finis. La connaissance de ces résultats sera essentielle à la précision du modèle et des simulations multi-domaines que nous allons établir.

Avant d'entamer les recherches de modélisation et simulation de ce prototype de micromanipulateur 2-D, nous allons d'abord présenter l'approche de conception que nous allons adopter dans cette thèse. En effet, la méthodologie de prototypage virtuel en passant par le langage VHDL-AMS de modélisation, et ses outils de développement, sont autant d'éléments qu'il faudra parfaitement maîtriser avant de choisir telle ou telle orientation de description. C'est ce que nous allons essayer de synthétiser dans le Chapitre 3 suivant.

Chapitre 3

Approche de modélisation et simulation par Prototypage Virtuel Fonctionnel

3.1 Introduction

Traditionnellement, dans le domaine microélectronique, nous retrouvons deux approches de conception [56] :

- *L'approche dite « descendante » (ou « top-down » en anglais) qui consiste à commencer par une abstraction du système de haut niveau, puis descendre progressivement aux descriptions de composants de base, comme le montre la figure 3.1.*
- *L'approche dite « ascendante » (ou « bottom-up » en anglais) qui consiste à commencer par les conceptions techniques de composants de base, ces « briques » seront ensuite utilisés afin de construire tout le système de base, comme le montre la figure 3.1.*

L'avantage de l'approche dite « descendante » est qu'elle permet de se concentrer uniquement sur la conception globale du système sans se soucier du choix technique inhérent à celle-ci. Ce qui ne sera pas le cas, inversement, avec l'approche dite « ascendante ». Pour la conception des systèmes d'une complexité élevée ces deux approches seront souvent utilisées conjointement.

L'avènement des technologies MEMS, et des nouvelles applications qui en ont découlé, n'a pas foncièrement modifié ces approches, bien au contraire. L'approche de conception des systèmes à base de MEMS s'est toujours inspirée de « l'existant » en matière de méthodologie et mode de fabrication des circuits intégrées propre à l'industrie de la microélectronique.

Cependant, comme nous l'avons introduit précédemment, les besoins inhérents aux MEMS requièrent des moyens en modélisation et simulation multi-domaines qui ne peuvent être résolus qu'à partir de méthodes et « d'outils langages » spécifiques [57, 58]. Dans ces domaines, la méthodologie de conception dite « Prototypage Virtuel Fonctionnel », que nous allons décrire

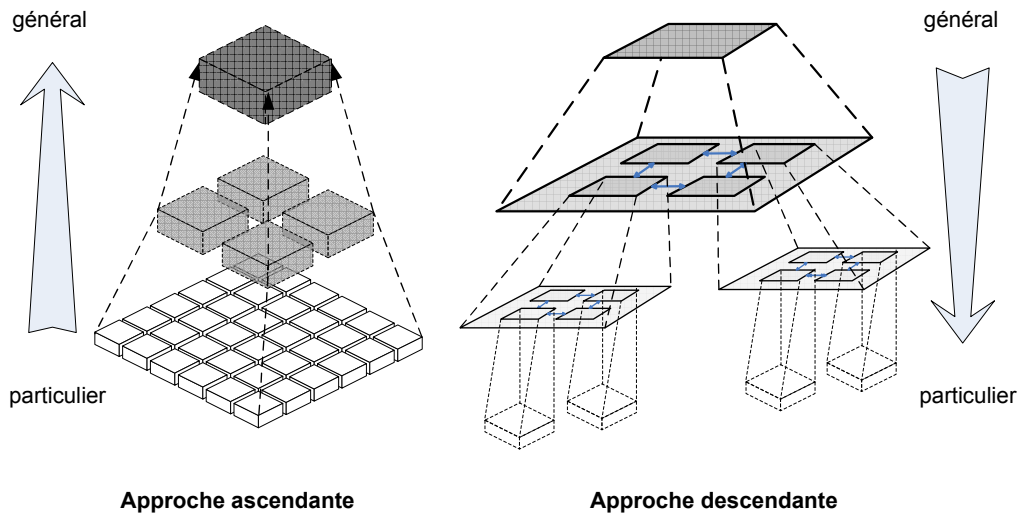


Fig. 3.1 – Approche de conception dite « descendante » et « ascendante »

dans la section suivante, semble la mieux adaptée et la plus prometteuse.

3.2 Prototypage Virtuel Fonctionnel

3.2.1 Méthode de conception dite du « Cycle en V »

3.2.1.1 Descriptif

Aujourd’hui, dans le domaine microélectronique et plus généralement dans celui de la conception de produits industriels, la méthode de conception utilisée se base sur le « cycle en V ».

Nous constatons que ce cycle de conception respecte l’approche « descendante » pour la conception et « ascendante » pour la validation. Il s’agit de deux phases avec plusieurs étapes de conception ou validation. La première phase est celle de la « conception » formant la branche « descendante » du « V », puis nous avons la phase de « validation » formant la branche montante du « V », comme le montre la figure 3.2.

La phase de « conception » débute par l’établissement du « cahier des charges ». On analyse ensuite le comportement fonctionnel du système dans l’étape « analyse fonctionnelle ». Une fois étudiées les caractéristiques globales du système, on descend au niveau de l’architecture dans l’étape « d’exploration architecturale » du système. Puis, on analyse, au niveau de l’étape « analyse des fonctions de base », les fonctions de base du système, lesquelles sont interprétées comme les composants élémentaires le constituant. Pour finir, les composants de base sont intégrés pour former le prototype du système, soit la spécification des composants au plus bas niveau que nous nommerons étape « prototype intégration »,

La phase de « validation » du « cycle en V » consiste aux tests et aux validations fonctionnelles et paramètres des sous-ensembles du système. Ces tests et validations sont effectués de

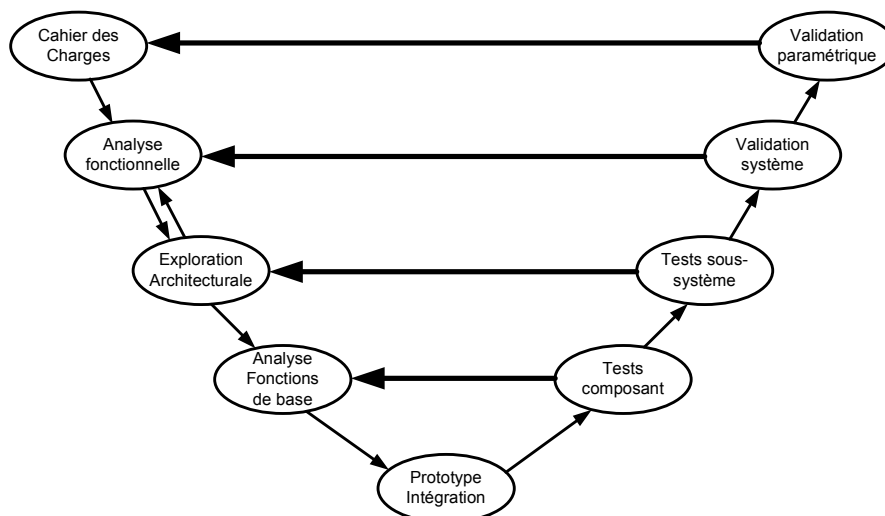


Fig. 3.2 – Cycle de conception dit « cycle en V » et ses boucles possibles de reprise

façon hiérarchique, à partir des composants de base pour remonter jusqu'au niveau système. On retrouve, sur la figure 3.2, ces étapes sous l'appellation « test composant », puis « test sous-système », suivies de « validation système » et finalement « validation paramétriques ».

A chacune des étapes de la phase de « validation » si une erreur est détectée, on retournera à l'étape correspondante dans la phase de conception, comme le montre la figure 3.2.

3.2.1.2 Inconvénients de la méthode

Les principaux inconvénients du cycle de conception dit « cycle en V » sont faciles à analyser. Plus une erreur de conception (non respect des spécifications) est découverte tard dans le cycle, plus la correction de cette erreur va coûter cher aux concepteurs.

En effet, dans le meilleur des cas, si une erreur est trouvée au niveau « composant », il suffira de refaire la dernière étape de la phase « conception ». En revanche, si une erreur apparaît au niveau « système », non seulement le cycle de conception risque d'être entièrement à reprendre mais aussi l'ensemble des tests effectués aux plus bas niveaux. Finalement, la perte sera double : en termes de coût et de temps de conception.

Comme on le voit dans cette introduction, le cycle de conception dit « cycle en V » ne répond pas parfaitement aux contraintes industrielles de plus en plus exigeantes. En effet, les demandes en gain de productivité sont telles aujourd'hui et dans le futur que l'on doit se diriger vers des cycles de conception intégrant des notions de « Prototypage Virtuel Fonctionnel » pour une conception « bonne du premier coup ». Ce sont ces nouveaux principes de conception que nous allons aborder à présent.

3.2.2 Le Prototypage Virtuel Fonctionnel

3.2.2.1 Descriptif

Le « *Prototypage Virtuel Fonctionnel* » est une nouvelle méthodologie de conception apparue récemment, formalisée par *Y. Hervé* [59], et qui permet de résoudre de très nombreux problèmes liés à la méthode classique de conception dite du « cycle en V ». Les principes du « *Prototypage Virtuel Fonctionnel* » se basent sur le renforcement à chaque étape du précédent « Cycle en V » par des modèles comportementaux décrits à un niveau d'abstraction bien choisi.

La figure 3.3 illustre cette approche en reprenant le « cycle en V » et en y apportant les modifications décrites. Ainsi, différentes étapes sont ajoutées tant au niveau de la phase de « conception » qu'à celui de la phase de « validation ». Le renforcement de toutes les étapes a pour but d'éviter les erreurs éventuellement intégrées lors de la phase de « conception » en essayant de les détecter le plus tôt possible dans la phase de « validation ». Ces nouvelles étapes, qui vont être rajoutées à la phase de « conception » du « cycle en V », sont données et détaillées dans ce qui suit :

1. **Description comportementale du niveau système :**

Avec cette étape, nous traduisons la définition du système en description de caractéristiques simulables.

2. **Modèle comportemental :**

Dans cette étape, un modèle décrivant le comportement global du système sera établi. Avec ce modèle de haut niveau, nous pouvons simuler le comportement du système de plus haut niveau. Les caractéristiques physiques des éventuels composants ne seront pas abordées.

3. **Modèle structurel comportemental :**

Dans cette étape, nous allons découper le « modèle comportemental » établi dans l'étape précédent en plusieurs sous-modèles indépendants. Ce modèle nous permet d'analyser la structure du système et les liaisons internes entre chaque sous-modèle. La description des sous-modèles du modèle est en général de haut niveau.

4. **Modèle des composants :**

Dans cette étape, les modèles décrivant les composants au niveau physique seront établis. Avec ce modèle de bas niveau d'abstraction, nous pouvons simuler le comportement physique de chaque composant.

De la même manière nous décrirons les étapes rajoutées à la phase de « validation » du « cycle en V » :

1. **Modèle structurel des composants :**

Dans ce modèle, les sous-modèles du « modèle structurel comportemental » de la phase de conception seront remplacés par les modèles des composants établis dans la dernière

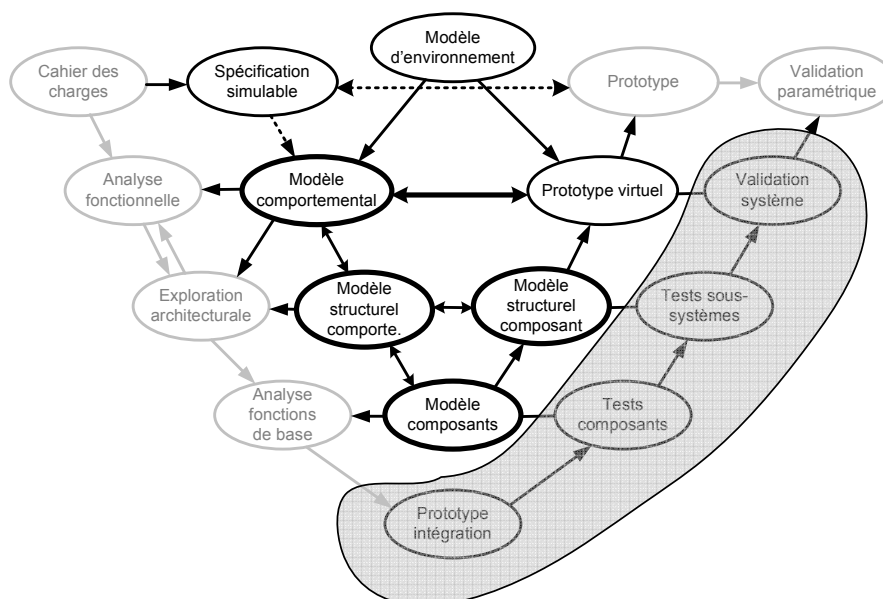


Fig. 3.3 – Cycle de conception du « Prototypage Virtuel Fonctionnel »

étape de la phase de conception. Avec ce modèle, nous pouvons connaître le comportement complet du système.

2. Prototypage Virtuel :

Avec le « modèle structurel des composants » établi dans l'étape précédent, nous pouvons construire déjà un « prototype virtuel » basé sur le prototype réel.

A noter que le renforcement des deux étapes nous permet d'effectuer tous les tests et validations de fonctionnement avec les modèles décrits dans l'approche de « Prototypage Virtuel Fonctionnel » à chaque étape et le prototype virtuel. Les étapes de validation propre au « cycle en V » pourront donc être préparées en amont et allégées, comme le montre la partie entourée en gris de la figure 3.3.

3.2.2.2 Avantages de la méthode

Les avantages apportés par l'approche de conception par « Prototypage Virtuel Fonctionnel » peuvent être résumés dans ce qui suit :

- Moins de risque de conception,
- Réduction de délais du projet,
- Réutilisation des résultats,
- Réduction du coût de prototypage,
- Augmentation de la qualité et de la fiabilité des produits.

Pourtant, la réalisation de ce cycle de conception dépend fortement des performances dues aux outils utilisés, lesquels doivent répondre aux caractéristiques suivantes :

- Unicité (le même langage pour toutes les étapes du cycle),
- Multi-abstractions,
- Multi-physiques,
- Normalisé.

3.2.3 Approche de modélisation

3.2.3.1 Coeur de conception

Le « Prototypage Virtuel Fonctionnel » est un cycle complet de conception. A partir de cette méthode, nous pouvons réaliser la conception d'un produit depuis l'établissement du cahier des charges jusqu'à la validation finale du produit « consommable ».

Le coeur de conception de cette méthodologie est l'élaboration de modèles dits « comportementaux » décrivant au plus près le comportement du système dans son environnement physique. La réalisation de ces modèles « comportementaux » se décompose en quatre étapes extraites du coeur du cycle de « Prototypage Virtuel Fonctionnel », comme le montre la figure 3.4. Ces quatre étapes sont :

1. L'étape « modèle comportemental »,
2. L'étape « modèle structurel comportemental »,
3. L'étape « modèles composants »,
4. L'étape « modèles structurel composant ».

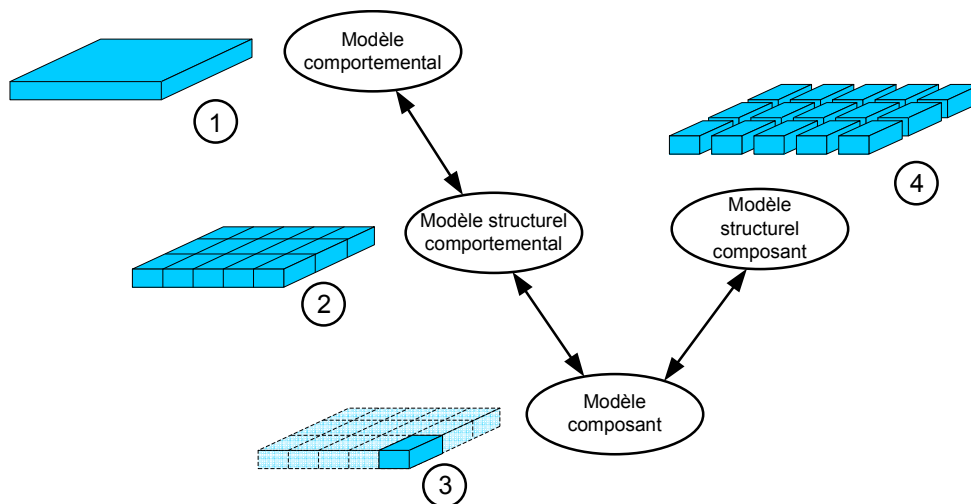


Fig. 3.4 – *Approche de modélisation intégrée au cycle de « Prototypage Virtuel Fonctionnel »*

3.2.3.2 Description des étapes de modélisation

1. **Modèle comportemental :**

Normalement, l'aspect d'un système complexe est multiple. Pourtant, nous considérons que son « modèle comportemental » est un bloc entier, nous simplifions les interactions de tous les paramètres et tous les signaux du système, comme s'ils étaient dans une seule et même structure. L'intérêt de ce modèle nous permet d'avoir une vue globale du système, nous ne nous intéressons qu'à la relation entre le système et l'environnement.

2. Modèle structurel comportemental :

Au niveau de cette étape, le « modèle comportemental » est dit structuré, c'est-à-dire qu'il peut être découpé en un certain nombre de composants selon leur fonctionnalité. Ensuite, le « modèle structurel comportemental » reprend ces composants pour les assembler en ne décrivant que les interactions qui les caractérisent. Les composants sont considérés comme de simples « boîtes noires » d'entrées-sorties dont on ne connaît que le rôle.

3. Modèles composants :

Dans cette étape, nous descendons au plus bas niveau de conception pour modéliser les fonctions de base de chaque composant. C'est-à-dire de remplir les « boîtes noires » des composants par les fonctions requises. Ces « modèles composants » sont les « briques » de base du modèle global. Pour établir ces modèles il est possible de faire appel aux outils métiers ad-hoc.

4. Modèle structurel composant :

Après avoir modélisé tous les composants, nous pouvons les intégrer dans le modèle structurel comportemental en remplaçant les composants « boîtes noires » par les « modèles composants ». Le nouveau modèle nous permet d'avoir non seulement la vue globale des comportements du système, mais aussi la fonction de chaque composant, ainsi que les interactions entre les composants.

Dans cette thèse, nous adopterons cette approche de modélisation. Nous montrerons qu'elle s'adapte parfaitement au domaine de description comportemental des MEMS et des applications qui en découlent.

En effet, s'il est parfois essentiel de trouver des modèles simples pour une première étude de simulation, ce qui pourrait correspondre à l'étape « *modèle comportemental* », il est aussi nécessaire de pouvoir décomposer jusqu'aux microstructures de base constituant les MEMS pour être sûr de leur fonctionnalité [60]. Cette étape correspond à l'étape « *modèle structurel composant* », laquelle permet de simuler de tels détails de structure tout en gardant l'avantage d'être simulé dans son environnement fonctionnel grâce à l'étape « *modèle comportemental* » qui l'a défini.

Dans la section suivante, nous allons présenter et analyser les avantages et les inconvénients du langage VHDL-AMS que nous adopterons pour décrire une telle approche de modélisation.

3.3 Les langages de modélisation descriptive

La réalisation du cycle de « Prototypage Virtuel Fonctionnel » que nous avons présenté précédemment nécessite un soutien très important du langage de modélisation et des outils concernés. Dans cette partie, nous allons présenter le langage VHDL-AMS, puis les autres langages de modélisation descriptive utilisés. Nous donnerons aussi un élément de comparaison de chacun de ces langages avec VHDL-AMS.

3.3.1 Le langage VHDL-AMS

3.3.1.1 Principes du langage

Le langage VHDL-AMS (VHSIC-Hardware Description Language - Analog and Mixed Systems) normalisé en décembre 1999 sous la référence IEEE 1076.1-1999 [61], a été développé comme une extension du langage VHDL (IEEE 1076-1993) [62].

Le VHDL-AMS permet la modélisation et la simulation de circuits et de systèmes logiques, analogiques et mixtes. En tant que sur-ensemble du VHDL, le VHDL-AMS profite des capacités de ce langage à modéliser des objets abstraits manipulant des signaux quantifiés à des moments discrets dans le temps. Aux instructions concurrentes et aux instanciations des modèles du VHDL, le VHDL-AMS ajoute les instructions simultanées qui permettent de manipuler des valeurs à temps continu transportées par les objets « Quantity », et le concept de synchronisation des noyaux de simulations numériques et analogiques utiles lorsque l'on veut forcer des points de simulation et réinitialiser des conditions initiales d'équations différentielles à des temps donnés. Les instructions simultanées mettent en jeu des équations différentielles non-linéaires implicites pouvant être simples, sélectives ou conditionnelles et qui relient les quantités.

Le VHDL-AMS dispose aussi de l'objet « Terminal » jouant le rôle de noeud de connexion pour le domaine analogique. L'objet « Terminal » est associé à une « Nature » qui définit un domaine physique. Afin que l'on puisse résoudre l'ensemble des équations différentielles, le critère de solvabilité doit être validé localement et les conditions initiales doivent être spécifiées [63, 64].

Pour l'instant, les modèles en VHDL-AMS ne peuvent pas utiliser des équations aux dérivées partielles. Il existe déjà de nombreuses recherches abordant ce problème. Par exemple, *Nikitin et al.* proposent d'appliquer une discrétisation spatiale pour contourner cette limite de VHDL-AMS [65].

3.3.1.2 Organisation d'un modèle VHDL-AMS

a. Unités de conception

« L'unité de conception » (design unit) est le plus petit module compilable séparément. VHDL-AMS offre cinq types d'unités de conception [66] :

- La déclaration d'entité (*entity declaration*) ;

- Le corps d'architecture (*architecture body*), ou plus simplement architecture d'une entité ;
- La déclaration de configuration (*configuration declaration*) ;
- La déclaration de paquetage (*package declaration*) ;
- Le corps de paquetage (*package body*).

Les trois premières « unités de conception » (*déclaration d'entité, architecture et déclaration de configuration*) permettent la description de l'aspect matériel d'un système, alors que les deux dernières (*déclaration et corps de paquetage*) permettent de grouper des informations pouvant être réutilisées pour la description de plusieurs systèmes différents.

La figure 3.5 reprend la disposition de ces unités de conception dans l'organisation globale du modèle VHDL-AMS [66].

Les trois unités de conception : déclaration d'entité, déclaration de configuration et déclaration de paquetage, sont qualifiées de *primaires* (*primary unit*, marquées par un **P** sur la figure 3.5), car elles décrivent une vue externe. Les unités primaires déclarent ce qui est accessible aux autres parties du modèle. Elles permettent le partage et la réutilisation de ressources (modèles et algorithmes). Elles ne contiennent que le minimum d'information nécessaire pour l'utilisation de ces ressources. Les deux autres unités de conception : architecture et corps de paquetage, sont qualifiées de *secondaires* (*secondary units*, marquées par un **S** sur la figure 3.5), car elles décrivent une vue interne particulière en fonction d'une vue primaire à laquelle elles sont « accrochées ».

Les unités secondaires regroupent des ressources seulement visibles par leur unité primaire correspondante (*déclaration d'entité pour l'architecture et déclaration de paquetage pour le corps de paquetage*). Elles contiennent des détails d'implémentation que l'on masque pour :

1. Éviter de surcharger l'utilisateur avec des détails inutiles.
2. Éviter des modifications malencontreuses de modèles et d'algorithmes de base. Plusieurs architectures peuvent être définies pour une même entité.

b. Exemple de programmation

Pour mieux connaître la structure d'un modèle en VHDL-AMS, un exemple est donné dans la figure 3.6. C'est un modèle de la résistance, qui a une valeur prédéfinie de $1\text{ K}\Omega$. Nous constatons qu'un modèle VHDL-AMS se compose d'une entité (**entity**) et une ou plusieurs architectures (il n'y a qu'une architecture dans l'exemple). Dans la partie **entity**, les bibliothèques sont appelées (**library**), les contenus sont précisés (**use**), les paramètres génériques sont définis (**generic**). Les ports (**port**) par lesquels les données sont reçues et envoyées sont aussi définis. L'entité décrit en fait l'interface du modèle avec le monde extérieur.

Dans l'architecture, on définit le comportement et/ou la structure du modèle. La première partie est une zone de déclaration dans laquelle les différentes classes d'objets sont déclarées, parmi lesquels, signal est utilisé pour le support des informations à événements discrets ; **quantity** est utilisé pour le support des informations à temps continu.

Cet exemple nous permet de découvrir les nouveautés apportées par le langage VHDL-AMS par rapport au VHDL, son sous-ensemble :

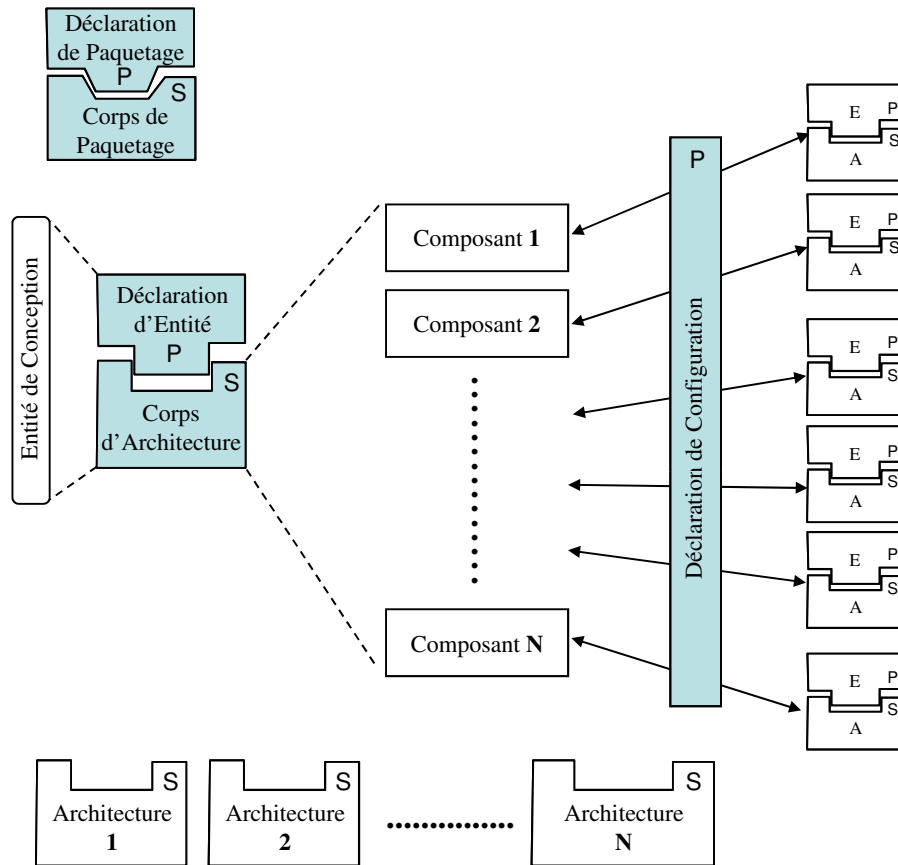


Fig. 3.5 – Unités de conception VHDL-AMS

```

library ieee;
use ieee.electrical_systems.all;

entity Resistance is
  generic (R : real := 1000.0);
  port (terminal a, b : electrical);
end;

architecture beh of Resistance is
  quantity V across I through a to b;
begin
  V == R * I;
end;

```

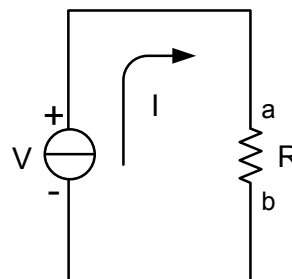


Fig. 3.6 – Exemple de modèle en VHDL-AMS : modèle d'une résistance

- **Terminal** : Ce nouvel objet de classe ne porte pas de valeur, il permet de nommer et distinguer les natures des noeuds de connexions.
- **Quantity** : C'est un objet nous permettant de manipuler des valeurs à temps continu, qui permet le mode de connexion *signal flow*.
- **Équations par les instructions simultanées** : Les liaisons entre les quantités sont exprimés par des équations différentielles temporelles, qui sont implémentées par les instructions simultanées symbolisées par ($==$). La position des quantités dans l'équation n'a aucune importance, c'est-à-dire qu'elles peuvent être utilisées de manières explicite ou implicite dans les équations.

Pour mettre en évidence les avantages de VHDL-AMS par rapport à d'autres langages de modélisation descriptive, nous allons étudier ces derniers dans la prochaine section en les comparant à VHDL-AMS.

3.3.2 Les autres langages de modélisation descriptive

3.3.2.1 Le langage Verilog-AMS

Le Verilog-AMS a été créé sous la tutelle d'Accellera (Organisation de normalisation EDA) afin de mettre en place les extensions analogiques mixtes du Verilog (IEEE-1364) [67]. La première version était Verilog-A LRM sortie en juin 1996 puis Verilog-AMS LRM en août 1998, puis Verilog-AMS LRM version 2.2 en novembre 2004.

Le langage Verilog-AMS permet de faire la description comportementale des systèmes analogiques et mixtes. Tout comme le VHDL-AMS, le Verilog-AMS peut être applicable aux systèmes électriques et non électriques. Ce langage permet de faire des descriptions de systèmes, en utilisant des concepts tels que les noeuds, les quantités de branche, et les ports. Les signaux de type analogique et numérique peuvent être présents dans le même module. D'ailleurs, Verilog-AMS fournit aussi la possibilité de modéliser des systèmes multi-physiques. Ce langage ne présente pas les mêmes hauts niveaux d'abstraction que VHDL-AMS. La norme est assez floue sur certains points rendant ses implémentations incompatibles.

3.3.2.2 Le langage MAST

Le langage MAST est un langage propriétaire développé en 1984 et complètement lié au simulateur SABER développé à l'origine par la société AnalogyTM et appartenant actuellement à la société SYNOPSYS [68]. Depuis plus de vingt ans, en tant que précurseur des langages de modélisation, le MAST a dominé le marché des langages de description de matériel à signaux et technologie mixte, de part le fait que l'outil SaberTM est implémenté notamment dans de nombreuses entreprises automobiles et aéronautiques. Ce langage propose néanmoins des mécanismes de modélisation obsolètes et non adaptés notamment pour la modélisation de systèmes numériques.

Le langage MAST est exclusif à l'outil SaberTM, et présente le défaut de ne pas être normalisé. De plus, il permet d'avoir au sein même du code du modèle des instructions pilotant directement le coeur du simulateur afin de contrôler la convergence (« control section »). La mise en commun de différents modèles ayant chacun un accès au coeur de simulation rend les projets d'une complexité extrême.

Ainsi l'inconvénient majeur de SaberTM, par rapport à notre vision de plate-forme coopérative, est son langage propriétaire, MAST, qui ralentit sa diffusion. Dernièrement, suite au rachat de l'outil par la société SynopsysTM et grâce à une réaction commerciale naturelle face à la montée en puissance de VHDL-AMS, une nouvelle initiative a été lancée. Il s'agit d'une proposition OpenMASTTM dont l'objectif est de rendre « open source » la grammaire du langage pour faciliter la naissance d'outils tiers.

Ce langage ne reprend cependant qu'une sous partie de la grammaire du MAST originel. L'autre réaction est la mise sur le marché de l'outil SaberHDL permettant de co-simuler du MAST et du VHDL-AMS. Cet outil n'est pas opérationnel à la date d'écriture de ce document.

3.3.2.3 Le langage Modelica

Modelica est un langage ouvert (non normalisé) dont le développement et la promotion sont organisés par l'association Modelica [69]. La première version de ses spécifications a été finalisée en septembre 1997 (version 1.0). La version la plus récente des spécifications est la version 2.2 (Mars 2005). C'est un langage de modélisation orienté objet qui permet de modéliser des systèmes complexes pouvant être multidisciplinaires (thermique, mécanique, hydraulique, électronique,...). Son écriture est plus accessible aux « informaticiens » qu'aux « physiciens/technologues ».

Les modèles y sont mathématiquement décrits à l'aide d'équations pouvant être algébriques, différentielles ou discrètes. Les algorithmes de résolution pour les systèmes équations y sont efficaces et permettent la manipulation de modèles complexes pouvant compter plusieurs milliers d'équations. Ce langage est utilisé dans des simulations de type hardware-in-the-loop (simulation mêlant prototype réel et prototype virtuel) pour des modèles simples (avec des contraintes fortes d'écriture).

Il est possible d'intégrer dans des modèles Modelica des informations concernant leur représentation graphique, facilitant leurs utilisations dans des outils différents à forte valeur ajoutée graphique (animations).

Contrairement à la simulation d'un modèle mixte VHDL-AMS qui met en jeu le couplage de deux simulateurs distincts dédiés respectivement à la partie discrète et à la partie continue, qui vont se synchroniser explicitement à certaines instants t (dépassement de seuil, évènements). Modelica voit quand à lui à tout instant l'ensemble des équations différentielles, algébriques et discrètes comme un seul et unique système d'équations ce qui rend la synchronisation automatique (mais hors de contrôle de l'utilisateur).

Cela met néanmoins en avant qu'il n'y a pas dans Modelica une prise en charge adaptée

du domaine discret comme le fait le VHDL-AMS. En effet les mécanismes proposés sont les mêmes que dans MAST, qui sont obsolètes. Sur le site officiel du langage les seules références bibliographiques pour parler de la description numérique en Modelica sont des références sur VHDL!

3.3.2.4 MATLAB

MATLAB est un interpréteur de commande pour accéder aux bibliothèques LINPACK et EISPACK créé par la société The MathWorksTM [70] en 1984. Il permet une manipulation simple des matrices, de tracer des fonctions et des données, d'implémenter des algorithmes, de créer des interfaces utilisateurs, et de s'interfacer facilement avec d'autres programmes. MATLAB est construit autour d'un langage communément appelé M-code. Le code est saisi dans une fenêtre de commande. La séquence de commandes peut alors être sauvegardée dans un script, ou encapsulée dans une fonction. MATLAB peut être complétée par de multiples outils appelés *plug-ins* ou *toolbox* :

- *Communications Toolbox*,
- *Control System Toolbox*,
- *Neural Network Toolbox*,
- *Optimization Toolbox*,
- *Robust Control Toolbox*,
- *Statistics Toolbox*,
- *System Identification Toolbox*,
- *Virtual Reality Toolbox*,
- *Excel Link*,
- *Simulink*.

L'outil SIMULINK proposé depuis 1992 permet de modéliser, simuler et analyser des systèmes dynamiques multi-domaines décrits sous forme *SIGNAL FLOW*. Son interface principale repose sur un diagramme de blocs graphiques, avec des bibliothèques de blocs prédéfinis permettant la modélisation système. L'outil profite de la puissance de calcul numérique offerte par l'environnement MATLAB. Il est principalement utilisé dans les applications de contrôle et dans la manipulation de signaux numériques (DSP).

Une de ses principales caractéristiques est que les connexions ne sont que de type « flot de signal » (*signal flow*). Dans ce cas de figure, les sorties d'un bloc ne sont pas influencées par les entrées d'un bloc connecté aux sorties (impédance de sortie nulle et d'entrée infinie). *S. Guessab* a réalisé une comparaison entre une modélisation sous MATLAB /Simulink et sous VHDL-AMS d'un composant piézoélectrique [71]. Il constate que le VHDL-AMS, de part son type de connexion conservatif (échange bidirectionnel de l'énergie) et son écriture des équations physiques, permet une modélisation plus aisée et plus directe. En termes de résultats de simulation, ils sont identiques pour une complexité de modélisation beaucoup plus grande en SIMULINK.

Pour être complet au niveau des possibilités d'utilisation de « Simulink » au sein d'un flot de conception système, il convient de citer certains logiciels pouvant être couplés à « Simulink » dans une optique de génération de code. Ainsi, en utilisant un autre produit de The MathWorks « Real-Time Workshop », « Simulink » est capable de générer du code C pour des systèmes embarqués à partir d'algorithmes créés sous « Simulink ». Avec le logiciel « Simulink HDL Coder », du code VHDL et Verilog synthétisables peuvent être générés à partir de « Simulink ».

La hiérarchisation des équipes de développement fait que chaque niveau du processus de conception système n'est pas conçu par les mêmes personnes, et chaque population a ses propres habitudes et ses propres outils « métier ». Par exemple, la description de plus haut niveau d'une chaîne de traitement de communication (exemple UMTS) sera développée par des personnes du traitement du signal, qui représentent généralement leurs algorithmes en code C ou en MATLAB. C'est le niveau en dessous qui sera traité par les électroniciens, qui travailleront alors en Verilog, VHDL ou SystemC.

3.3.2.5 Tableau comparatif et bilan

Le tableau 3.1 permet de récapituler et comparer les principales caractéristiques des cinq principaux candidats de langage de modélisation et outil : VHDL-AMS, Verilog-AMS, Modelica, MAST et MATLAB.

D'après les études et les comparaisons faites entre les langages de modélisation, il en ressort que les mécanismes de modélisation de la norme VHDL-AMS (multi-domaines, multi-abstractions, multi-architectures) apparaissent les mieux adaptés à notre méthodologie de conception par « Prototypage Virtuel Fonctionnel ». Ce langage de haut niveau permet d'écrire un cahier des charges abstrait et simulable. Une architecture de système peut être faite de manière fonctionnelle en associant des modèles existants ou à créer. Les modèles peuvent être décrits sous forme comportementale ou structurelle.

Le fait d'être une norme, le VHDL-AMS autorise l'échange entre collaborateurs ou de fournir à des clients des modèles de haut niveau dans le cadre d'une bibliothèque IP (**Intellectual Property**). Il est aussi possible de décrire les composants à très bas niveau en écrivant sous forme d'instructions simultanées les équations différentielles issues des études physiques. On peut donc ajouter dans les modèles des imperfections et ainsi voir l'influence sur les performances de la modification d'un paramètre.

VHDL-AMS permet de modéliser l'environnement du système, optimisant ainsi son étude et sa mise au point. Le VHDL-AMS prend en charge la conception mixte et permet de mettre en jeu plusieurs domaines physiques. Ainsi une modélisation peut prendre en compte les phénomènes électriques, optiques, mécaniques, thermiques et leurs couplages. Dans ce domaine, de nombreux modèles ont été conçus avec VHDL-AMS. Par exemple, un modèle de transformateur électromagnétique [73], un modèle de capteur piézo-céramique ultrasonique [74], un capteur « bluetooth » [75] mais aussi des véhicules hybrides, des moteurs électriques, des mécanismes bio-

	VHDL-AMS	Verilog-AMS	Modelica	MAST	MATLAB
Norme	IEEE 1076.1-1999	Recommandation	Non Définition ouverte	Non	Non
Présentation	Extension du VHDL (IEEE 1076)	Extension de Verilog (IEEE 1364)	Langage de modélisation orienté objet	Langage propriétaire lié à l'outil SABER	Logiciel calcul numérique + Simulink
Modularité	Entité Active Multi-Architecture	Mono-Architecture	Mono-Architecture	Mono-bloc	Mono-Architecture
Équations algèbre-différentielles	Oui NL-implicite ordre N	Oui Forme explicite ordre 1	Oui Forme implicite ordre 1	Oui Forme explicite ordre 1	Oui Forme explicite ordre 1
Multi-disciplines	Oui	Non	Oui	Oui	Oui
Conservatif / Flot de signal	Oui/Oui	Non/Oui	Oui/Oui	Oui/Oui	Oui/Oui
Multi-abstraction	Oui	Non	Oui	Oui	Non
Gestion des discontinuités	Oui Instruction Break	Oui	Oui/automatique Mécanisme daté	Oui Control Section	Non
Dépendance à un outil	Non	Oui	Oui	Non	Non

Tab. 3.1 – Comparatif de cinq langages de modélisation mixte [72]

chimiques, des batteries sous forme électrochimiques. Voir le site www.systemsvip.com [76].

Un autre atout du VHDL-AMS est de proposer une approche multi-abstractions permettant ainsi d'associer des modèles de haut niveau et des modèles de bas niveau dans le même modèle global ayant une approche structurelle. Ainsi, on peut détailler certaines parties plus finement, alors que d'autres parties sont modélisées de manière plus abstraite, requérant ainsi un temps de calcul inférieur [6, 77] à une modélisation homogène à bas niveau.

Un des intérêts du VHDL-AMS est de construire une bibliothèque de modèles génériques réutilisables afin d'augmenter la productivité. A la suite de sa normalisation en 1999, de nombreux projets ont été mis en place autour de ce langage. Les laboratoires de recherches et les industriels ont unis leurs efforts afin de développer les bibliothèques de modèles dans ce langage. Ces efforts ont aussi permis d'améliorer constamment les outils dédiés à ce langage et notamment les simulateurs de modèle.

Les critères importants des concepteurs sont en effet pour ces outils : la couverture de la norme, une interface conviviale, une gestion des bibliothèques de modèles efficace, des simulations performantes en termes de précision et de temps de mise en oeuvre.

3.4 Outils de simulation VHDL-AMS

Actuellement, il existe plusieurs simulateurs pour la simulation du VHDL-AMS. Aucun d'entre eux ne supporte la norme à 100%, mais chacun s'en approche plus ou moins. Nous allons présenter trois des plus utilisés.

3.4.1 Environnement de simulation VHDL-AMS

La figure 3.7 illustre l'environnement de simulation type de VHDL-AMS et les différentes phases « d'édition », « d'analyse », « d'élaboration » et « d'exécution » liées au langage [66].

L'interface graphique peut se réduire à un simple éditeur de texte. Les outils CAO du marché utilisent en plus leur éditeur de schémas pour générer automatiquement le squelette d'un modèle VHDL-AMS, c'est-à-dire au moins la déclaration d'entité avec ses ports et un corps d'architecture minimum. Des outils plus avancés permettent de décrire le comportement du système à modéliser sous la forme de machines d'états, de chronogrammes ou de tables de vérité.

L'analyseur (ou compilateur) vérifie la syntaxe d'une description VHDL-AMS. Il permet la détection d'erreurs locales, qui ne concernent que l'unité compilée. Plusieurs techniques d'analyse sont actuellement utilisées par les outils du marché. L'approche compilée produit directement du code machine, ou, dans certains cas, du code C++ qui sera lui-même compilé. L'objet binaire est alors lié au code objet du simulateur. Cette approche réduit le temps de simulation au détriment du temps d'analyse. L'approche interprétée transforme le code source en un pseudo-code qui est interprété par le simulateur. Cette approche réduit le temps d'analyse au détriment du temps

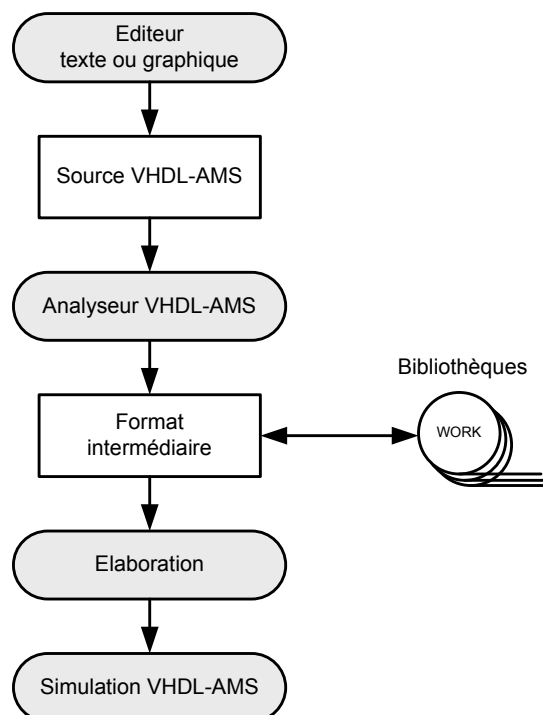


Fig. 3.7 – *Environnement de simulation VHDL-AMS*

de simulation.

Chaque concepteur possède une bibliothèque de travail (**working library**) de nom logique WORK (le nom est standard) dans laquelle sont placés tous les modèles compilés. Le lien du nom logique avec l'emplacement physique de la bibliothèque dépend de l'outil de simulation ou de synthèse utilisé.

Il est aussi possible de faire référence, en mode de lecture seule, à d'autres bibliothèques, des bibliothèques de ressources, contenant d'autres modèles ou des utilitaires. Plusieurs bibliothèques peuvent être actives simultanément. Chaque bibliothèque contient une collection de modèles mémorisés dans un format intermédiaire. Elle contient également un certain nombre de relations et d'attributs liant, si nécessaire, les différents modèles entre eux.

La phase d'élaboration consiste en une construction des structures de données et permet la détection d'erreurs globales, qui concernent l'ensemble des unités de la description. Cette phase est normalement exécutée en arrière-plan avant la simulation proprement dite.

Le simulateur calcule comment le système modélisé se comporte lorsqu'on lui applique un ensemble de stimuli. L'environnement de test peut également être écrit en VHDL-AMS : il peut être lui-même vu comme un système définissant les stimuli et les opérations à appliquer aux signaux de sortie pour les visualiser (sous forme texte ou graphique). Le simulateur permet aussi le déverminage (debugging) d'un modèle au moyen de techniques analogues à celles proposées pour les programmes écrits en Pascal, C ou Ada : simulation pas à pas, visualisation de variables, de signaux, modification interactive de valeurs, etc. Il faut noter que ce genre de vérification s'ap-

plique difficilement à un modèle analogique. En effet : quelle est la signification d'une simulation pas à pas des lignes de code d'un ensemble d'équations simultanées ?

3.4.2 Principaux outils de simulation

3.4.2.1 hAMSter et Simplorer (Ansoft®)

hAMSter est un compilateur/simulateur du langage VHDL-AMS librement disponible. Ce n'est pas un outil industriel. Son principal avantage est sa simplicité d'utilisation, il permet à l'utilisateur de prendre en main très rapidement les démarches essentielles du travail avec le VHDL-AMS. Il n'a pas besoin d'une configuration élevée d'un ordinateur pour fonctionner. Toutefois, il montre aussi quelques inconvénients. Le plus marquant est que le logiciel présente beaucoup de *bugs*. Les tout premiers modèles développés dans cette thèse ont été réalisés avec ce logiciel.

hAMSter a finalement été absorbé par AnsoftTM, un leader dans le développement de logiciel pour la conception et la simulation électronique de haute technologie [78]. Un des produits phares d'AnsoftTM est SimplorerTM dont le premier moteur VHDL-AMS a été construit sur hAMSter. C'est une suite logicielle composée de plusieurs modules et destinée essentiellement à simuler le fonctionnement de systèmes comportant des circuits électriques ou électroniques, des organes mécaniques ou des dispositifs de contrôle. Ses domaines d'application sont très vastes, avec une prédilection pour les systèmes ou processus industriels.

Lors des premiers travaux de développement des modèles, les problèmes rencontrés le plus souvent sont dûs aux *bugs* du logiciel. A titre d'exemple, quand on décrit un objet qui rebondit sur le sol, il est arrivé que le résultat montre que la hauteur de l'objet est négative, c'est-à-dire d'entrer dans le sol. La cause de cette erreur vient d'un calcul de pas et de détection de choc pas assez précis.

3.4.2.2 ADVance-MS (ANACAD)

C'est un outil développé par ANACAD, société française rachetée par MENTOR-Graphics intégré dans la chaîne de conception de cette société [79]. Il fonctionne sur les stations SUN et HP. C'est un outil issu et conçu pour le monde de l'électronique intégrée. Aucun effort n'est fait par l'éditeur pour augmenter la couverture de la norme pour accéder à des besoins de modélisation de niveau système. Le client principal est ST Microelectronics et seules des demandes pilotent la « roadmap » des développeurs.

Par exemple au niveau de la couverture de la norme, il y a beaucoup de lacunes dans les premières versions. A titre exemple, Initialisation, « break » et « generate » ne sont pas supportés. Depuis quelques années, cet outil est utilisé régulièrement dans la conception des MEMS, nous pouvons citer une étude sur une microcellule de vibration mécanique par Schlegel [80].

3.4.3 Simulateur SMASH (Dolphin-integration®)

SMASHTM est en ce moment le simulateur du langage VHDL-AMS le plus complet sur le marché. Il s'approche d'une couverture totale de la norme, assez loin devant les autres simulateurs [81]. Dans sa version de base il ne dispose pas d'éditeur graphique.

Une capacité très intéressante de l'outil SMASHTM est sa capacité multi-langage. Il est compatible avec SPICE, Verilog-HDL, VHDL, ABCD (internal C-language), C et VHDL-AMS et peut être interfacé avec MATLAB/SIMULINK. A partir de la version SMASH 5, les utilisateurs ont plus de possibilités de travailler en mélangeant les différents langages et les niveaux de description. Il permet surtout d'ajouter dans un modèle un autre modèle développé dans un autre langage.

Cette capacité est bénéfique pour le développement d'un projet complexe. Nous pouvons réutiliser un modèle déjà développé sous un autre langage sans le convertir. Cela réduit le temps de développement, et évite les éventuelles erreurs commises dans les conversions. Dans la perspective de notre projet actuel, il est prévu d'intégrer des modèles développés en langage C.

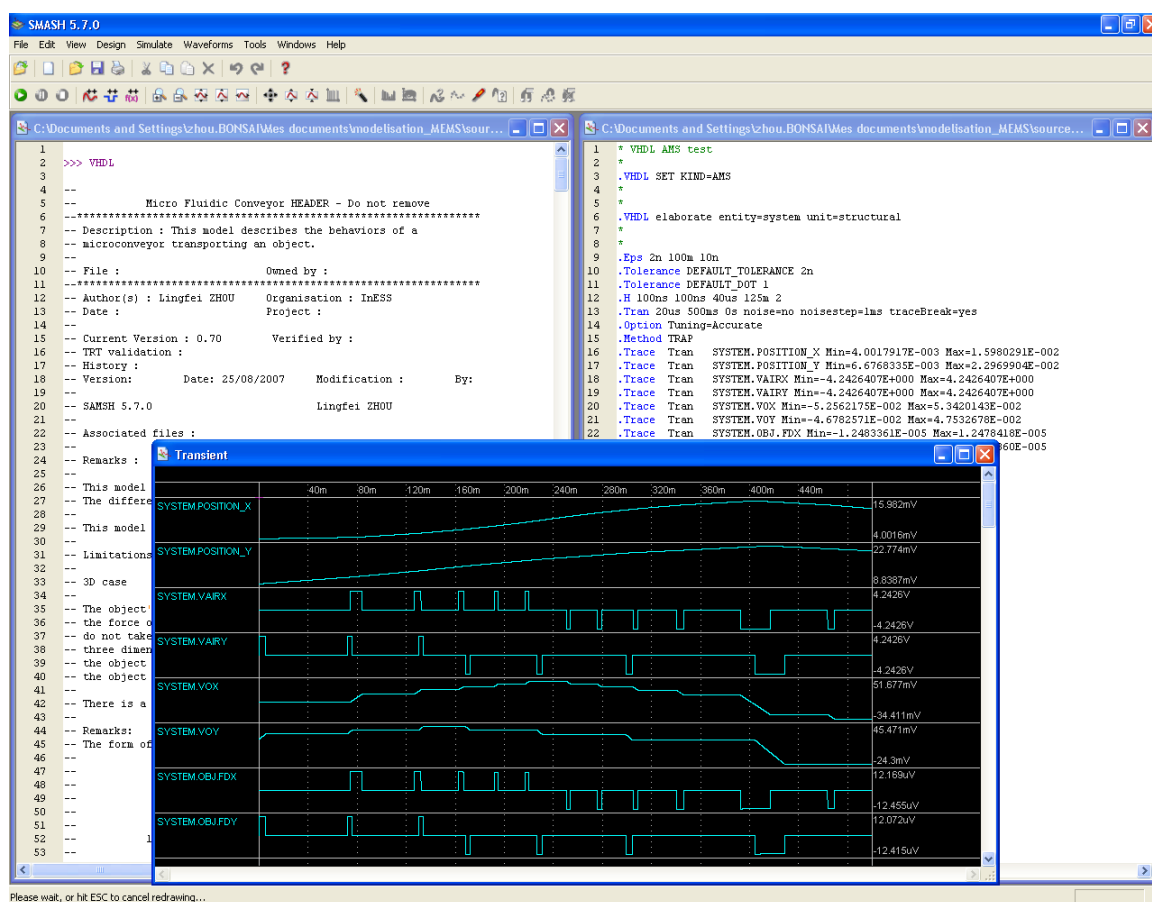


Fig. 3.8 – Environnement graphique du logiciel SMASHTM

La figure 3.8 donne un exemple d'une fenêtre de travail principale de SMASH. Deux fenêtres

côte-à-côte dans le fond affichent deux types de fichier de base : « .nsx » et « .pat ». Ces deux fichiers comprennent respectivement la description du modèle, et la description des directives de contrôle du compilateur et du simulateur. Dans la figure, la fenêtre à fond noir est le « grapheur » qui présente les résultats de simulation.

3.4.3.1 Autres outils

Saber® HDL : Saber® HDL est un produit de Synopsys®. Il s'agit d'un simulateur multi-technologies pour les systèmes mixtes. Il est indépendant des langages utilisés. Il peut supporter la combinaison des langages HDL et la description SPICE. Actuellement, il supporte aussi le langage VHDL-AMS et MAST. Saber®HDL fonctionne sur Sun Solaris 2.6.8, Windows NT 4.0/2000 et RedHat Linux 7.2 [82].

VHDL-AMS Wizard : VHDL-AMS Wizard est un outil développé par Ansoft® et intégré dans le logiciel Simplorer®. Il s'agit d'un générateur automatique de modèle en VHDL-AMS. Un éditeur avec la coloration syntaxique est mis en place. On peut importer les codes VHDL-AMS. Il existe aussi des bibliothèques des composants de base [78].

3.4.4 Contexte de thèse

Étant donné que l'émergence du langage VHDL-AMS est relativement récente, les outils de support comme les simulateurs ne sont ni nombreux, ni performants depuis le début. Il a fallu quelques années pour que les différents éditeurs de logiciels améliorent la performance de leurs produits.

Durant le développement des modèles dans le cadre de la thèse, les trois simulateurs cités dans les sections précédentes ont tous été utilisés. Nous avons finalement choisi SMASH™ comme l'outil principal pour la modélisation de la « Surface Distribuée à base de MEMS ». Les raisons sont les suivantes :

- SMASH est assez facile à utiliser ;
- Le simulateur SMASH est performant ;
- SMASH supporte des jeux d'équations de très grande taille ;
- SMASH offre de nombreuses possibilités dans la matière de la simulation ;
- Par rapport à ADV-MS, SMASH couvre mieux la norme ;
- SMASH présente de bonnes perspectives dans la co-simulation avec d'autres langages de modélisation ;
- La société Systems'ViP avec qui nous sommes en contact étroit a une très grande expertise et développe des outils tiers autour de ce noyau.

3.5 Conclusion

Dans ce chapitre, nous avons abordé trois sujets : la méthodologie de conception, le langage de modélisation choisi pour la réalisation de cette méthodologie de conception et l'outil de développement.

Dans la première partie, nous avons présenté la méthodologie de conception classique : Conception en V classique et nouvelle méthodologie formalisée par *Yannick Hervé* : le Prototypage Virtuel fonctionnel. Par rapport à l'ancienne méthode de conception, le prototypage virtuel améliore l'efficacité de la construction du système en renforçant l'étape de simulation. Cette modification apporte non seulement la réduction du temps et du coût de développement, mais aussi une bien meilleure fiabilité.

Dans la deuxième partie, nous avons d'abord présenté brièvement les cinq langages de modélisation mixte les plus présents sur le marché : Verilog-AMS, MATLAB, Modelica, MAST et VHDL-AMS, puis donné notre choix après une comparaison entre les caractéristiques des cinq candidats : VHDL-AMS. Ce langage permet la modélisation et la simulation de circuits et de systèmes logiques, analogiques et mixtes. Ses multiples avantages correspondent aux besoins du développement de notre système.

Dans la dernière partie, nous avons présenté plusieurs outils couramment utilisés dans la modélisation. Nous avons choisi SMASH comme l'outil de développement à cause de sa couverture de la norme et de sa capacité multi-langages.

Chapitre 4

Modèle comportemental

4.1 Approche de description

4.1.1 Description générale

Abordons dans ce chapitre la première étape de modélisation en VHDL-AMS de la matrice de microactionneurs pneumatiques dans son environnement fluide. Nous débuterons par la « modélisation comportementale » du dispositif et de son milieu fonctionnel, respectant ainsi l'approche du cycle de description de prototypage virtuel présentée précédemment.

On rappelle que la « modélisation comportementale » se doit d'être indépendante de l'architecture du système en explicitant seulement le fonctionnement par l'écriture d'un modèle global, c'est-à-dire de représentation mathématique associant des variables (appelées variables d'états) et des équations matérialisant les relations entre ces variables. De plus, à la différence d'une approche purement fonctionnelle, la description comportementale inclut des entrées/sorties et les lois de conservation de l'énergie, ce qui nécessite la définition, pour chaque accès du système, de quantités d'effort et de flux dont nous reparlerons.

Reprenons d'abord, avec la figure 4.1, le descriptif du système de micromanipulation plané basé sur une matrice de microactionneurs pneumatiques que nous allons modéliser en lui associant des fonctions fluidiques à écoulement d'air.

Sur la face-avant de la matrice présentée sur la figure 4.1, plusieurs rangées d'orifices sont alignées régulièrement. Les orifices sont conçus de façon soit horizontale, soit verticale. A chacun d'eux est associé, en face-arrière, un microactionneur électrostatique correspondant à une microvalve actionnée dans le plan dans deux directions.

Ainsi sera constituée la matrice de microactionneurs pneumatiques, laquelle se présentera finalement sous la forme d'un champ de force constitué de centaines de jets d'air. Ce réseau sera commandé de manière individuelle ou collective (par rangées) pour la micromanipulation sans contact et à deux dimensions d'un objet glissant (lévitant) sur sa surface.

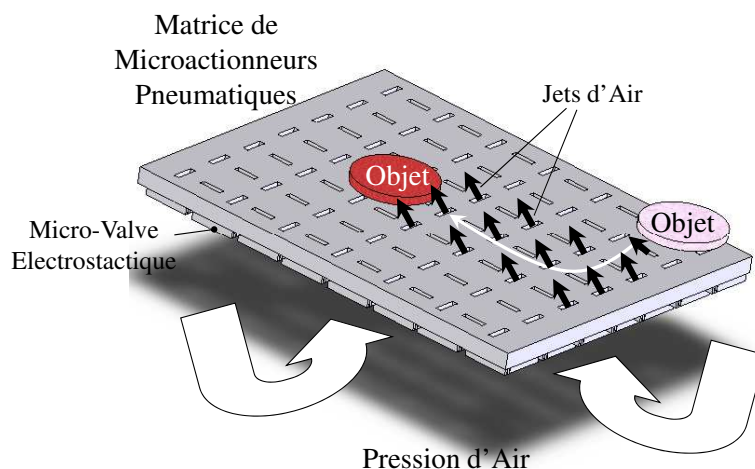


Fig. 4.1 – Description de la matrice de microactionneurs pneumatiques dans son environnement fluïdique à écoulement d'air

Pour décrire le comportement de la matrice, nous avons tout d'abord élaboré un modèle simpliste du dispositif comme l'illustre les deux représentations de la figure 4.2, correspondant respectivement aux vues de dessus (figure 4.2a) et de côté (coupe entre A et A' de la figure 4.2b) du dispositif de micromanipulation.

4.1.2 Description en sous-modèles

Au regard de ces figures nous allons définir les différentes règles de comportement des quatre sous-modèles structurels et fonctionnels du système :

- Le sous-modèle « microactionneur pneumatique »,
- Le sous-modèle « matrice de microactionneurs pneumatiques »,
- Le sous-modèle « objet »
- Le sous-modèle « environnement fluïdique à écoulement d'air ».

Détaillons à présent ces règles de description comportementale du système en entier.

4.1.2.1 Sous-modèle « microactionneur pneumatique »

La description comportementale du « microactionneur pneumatique » présentée sur la figure 4.2 se résume aux 4 règles structurelles et fonctionnelles suivantes :

- L'aire occupée par le microactionneur pneumatique est de forme carrée,
- L'orifice est de forme rectangulaire et se place au centre du microactionneur,
- Le microactionneur possède trois états :
 - État au repos : c'est l'état où la micro-valve n'est pas actionnée. Dans cet état, l'écart entre la micro-valve et l'orifice supérieur laisse des fuites d'air sur les côtés de ces éléments. Ainsi, le microactionneur génère une force de direction verticale et de sens vers le haut qui permettra la lévitation permanente de l'objet.

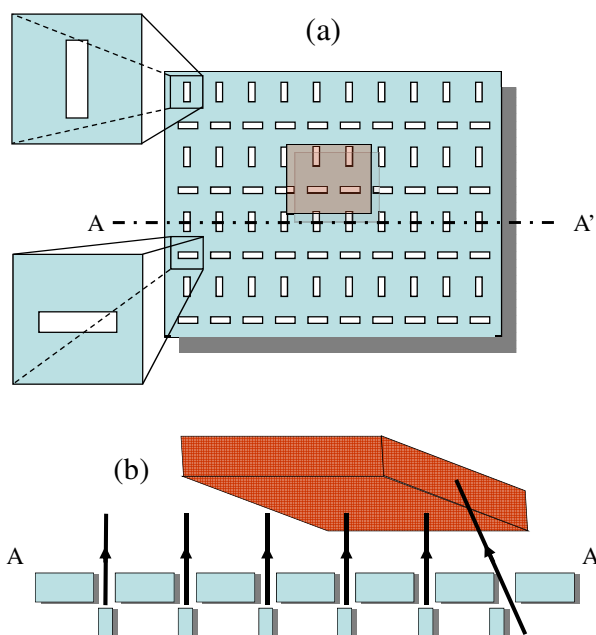


Fig. 4.2 – Description de la matrice de microactionneurs pneumatiques. (a) Vue de dessus. (b) Vue de côté.

- État actif 1 et 2 : c'est l'état où la micro-valve est actionnée. Dans cet état, la micro-valve est déplacée sur la droite ou sur la gauche (état actif 2), générant un jet d'air épousant la fente créée par l'écart entre la micro-valve et l'orifice.
- Le microactionneur peut être placé à la perpendiculaire sur le plan horizontal et ainsi permettre à ce dernier de diriger un jet d'air dans deux nouvelles directions avec un angle de 90° aux précédentes.

4.1.2.2 Sous-modèle « Matrice de microactionneurs pneumatiques »

Au vu des figures 4.1 et 4.2, la description comportementale de la matrice de microactionneurs, qui correspond à la surface active, est donnée par les 5 règles structurelles et fonctionnelles suivantes :

- La surface active est composée d'une matrice de microactionneurs pneumatiques, tels qu'ils sont dans la description précédente,
- Les microactionneurs sont placés rangée par rangée,
- Chaque rangée se compose d'un arrangement linéaire de microactionneurs projetant tous un jet d'air dans la même direction,
- Deux types de rangées sont placés alternativement,
- Les interfaces entre les rangées sont identiques.

4.1.2.3 Sous-modèle « Objet »

La description comportementale de l'objet, qui doit être manipulé, est représentée sur les figures 4.1 et 4.2 et peut être donnée par les 2 règles structurelles et fonctionnelles suivantes :

- L'objet est carré, plat et son corps est homogène (densité constante),
- La rotation de l'objet est négligée.

4.1.2.4 Sous-modèle « Environnement fluide à écoulement d'air »

La description de ce sous-modèle fait l'objet de la section suivante sur les « éléments de mécanique des fluides ».

4.2 Éléments de mécanique des fluides

4.2.1 Introduction

D'après *P. Chassaing* [83], "La mécanique des fluides est une discipline vaste et diversifiée, tant par les champs scientifiques qu'elle recouvre que par ses domaines applicatifs", et *Comollet* [84], "D'un point de vue thématique, la mécanique des fluides procède d'une analyse macroscopique de tous les mécanismes de transferts diffusifs par agitation moléculaire. Désormais l'assise scientifique de cette discipline est à rechercher autour d'un nouveau pôle barycentrique entre physique, mécanique, thermodynamique des processus irréversibles, thermique, chimie. Pour s'en convaincre, il suffit de prendre l'exemple de « l'écoulement » dans une turbomachine avec coexistence d'aspects diphasiques, réactifs, thermiques... où la connaissance du mélange est souvent l'une des clés de la maîtrise de tels équipements, renvoyant à une mécanique des fluides fort complexe".

Nous montrons à l'évidence qu'il ne saurait être question de couvrir d'un seul élan ce vaste champ de la mécanique des fluides pour introduire les notions élémentaires dont nous aurons besoin pour la modélisation de notre système de surface active pneumatique à base de MEMS distribués. Nous limiterons nos recherches aux seuls « mouvements des fluides parfaits » d'où nous étudierons les notions « d'efforts exercés par un fluide sur un solide » en nous basant sur les principes fondamentaux exprimées par *Y. A. Cengel* et *J. M. Cimbala* dans leur ouvrage de référence « Fluid Mechanics - Fundamentals and Applications » [85].

4.2.2 Définitions d'un écoulement fluide

L'introduction de marqueurs (fumée dans le cas des gaz, colorant pour les liquides), permet d'observer des différences importantes dans le comportement des écoulements des fluides [86]. Dans certains écoulements, les particules marquées diffusent très lentement c'est-à-dire s'écartent peu les unes des autres, les différentes couches (lamelles) glissent les unes par rapport aux autres

sans se mélanger : l'écoulement est dit « laminaire ». Au contraire dans d'autres écoulements les particules marquées s'éloignent très rapidement de manière « aléatoire, irrégulière, dans toutes les directions » les unes des autres, on ne retrouve plus de trace de marquage significative très près de l'endroit où le marqueur a été introduit : l'écoulement est dit « turbulent ».

Dans le cas d'un écoulement laminaire, les couches de fluide s'entraînent collectivement par friction visqueuse, sans se mélanger. A l'opposé, les écoulements turbulents sont le siège de mélange entre les couches rapides et les couches lentes, par tourbillonnement.

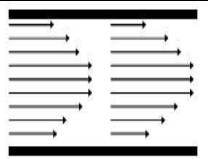

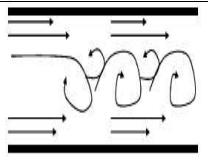

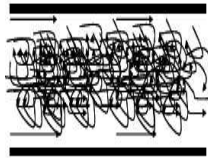

Le fluide n'étant pas assez visqueux, les effets d'inertie prévalent. Même si une partie des variables caractéristiques varie fortement en régime turbulent, il est possible d'analyser ces écoulements en considérant les variables comme la somme d'une donnée constante et d'une donnée fluctuante.

Nombre de Reynolds : Il caractérise l'importance des effets visqueux par rapport aux effets d'inertie du fluide en mouvement. Calculons le rapport entre les deux quantités et normalisons les grandeurs :

$$Re = \frac{\rho V L}{\mu} = \frac{V L}{\nu} \quad (4.1)$$

avec : (V) la vitesse moyenne de l'écoulement ; (μ) la viscosité cinématique du fluide ; (L) une longueur caractéristique ; (ρ) la masse volumique ; (ν) la viscosité cinématique.

Une loi empirique fondée sur les observations de Reynolds permet de prévoir le comportement d'un écoulement par le calcul de son nombre de Reynolds caractéristique. Les études empiriques donnent les résultats suivants, tableau 4.1 :

Nombre de Reynolds	Type d'écoulement	Écoulement d'un fluide visqueux dans une conduite	Visualisation de l'écoulement (traceurs colorés)
$Re < 2000$	Laminaire		
$2000 < Re < 6000$	Transitionnel		
$Re > 6000$	Turbulent		

Tab. 4.1 – Caractéristiques des écoulements visqueux en friction du nombre de Reynolds

4.2.3 Efforts exercés par un écoulement sur un solide

4.2.3.1 Phénomènes physiques d'un écoulement fluïdique

Comme nous l'observons quotidiennement dans la pratique, l'écoulement d'un fluïde d'air sur un corps solide se produit très fréquemment et est responsable de nombreux phénomènes physiques agissant sur les automobiles, l'habitat, les arbres, les lignes de hautes tensions, etc. Il agit également, mais de manière différente, sur les ailes des avions, la pluie (*upward draft of rain*), la neige, les particules de poussières, etc. La figure 4.3 représente quelques exemples de corps solides où s'appliquent des efforts exercés par l'écoulement de fluïde d'air (avion, montgolfière, hélicoptère, deltaplane).



Fig. 4.3 – Exemples de corps solides où s'appliquent les forces de traînée et de portance (avion, montgolfière, hélicoptère, deltaplane)

Développer une meilleure compréhension de ce type de phénomène fluïdique dit « fluïde extérieur » est un point essentiel dans la conception de nombreux systèmes d'ingénierie tels que l'aviation, l'automobile, l'habitat, les navires, les sous-marins et tous types de systèmes à turbine. Dans le domaine des écoulements d'air, les récents modèles de voiture, par exemple, doivent être conçus en insistant particulièrement sur l'aérodynamisme. Il en a résulté des réductions significatives dans la consommation d'énergie thermique mais également du bruit, et dans une considérable amélioration dans la conduite.

A noter que l'on retrouve ces mêmes phénomènes physiques dans l'écoulement des fluïdes liquides.

On notera aussi que si parfois c'est le fluïde qui se déplace vers un corps stable (comme

le vent soufflant sur un building), c'est aussi le corps qui peut aller à l'encontre d'un fluide dit « passif » (comme la voiture se déplaçant contre l'écoulement d'air). Ces deux processus apparemment différents sont en réalité équivalents. En effet, il importe peu de savoir quel est le mouvement absolu entre le fluide et le corps solide. De tels mouvements, référencés comme « fluide sur corps » ou « fluide extérieur », sont analysés de la même manière en fixant, par convention, les coordonnées du système sur le corps solide.

4.2.3.2 Écoulements fluidiques à deux et trois dimensions

La forme d'un corps solide à une profonde influence sur les effets de fluide sur un corps et sur les champs de vitesse. L'écoulement d'un fluide sur un corps est dit à deux dimensions quand le corps est très long et de section constante, et quand le fluide s'exerce de façon normale sur le corps solide. Le vent soufflant sur un long tube de façon perpendiculaire à son axe est un exemple de fluide à deux dimensions, comme l'illustre la figure 4.4(a). L'approche idéale du fluide à deux dimensions est appropriée quand le corps est suffisamment long pour négliger les effets de bords et quand le fluide est uniforme.

Un autre phénomène se produit aussi quand le corps développe des symétries rationnelles par rapport à un fluide qui doit être axisymétrique. Une balle traversant l'air est un exemple de fluide axisymétrique. La vitesse dans ce cas varie avec la distance axiale (x) et la distance radiale r , comme le montre la figure 4.4(b).

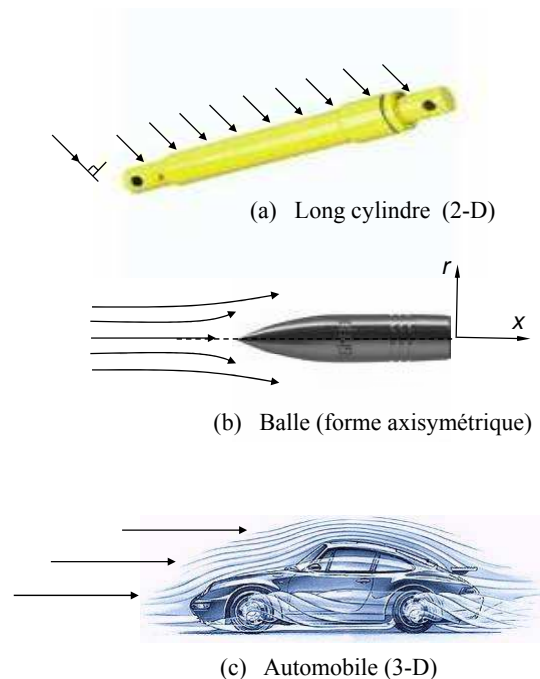


Fig. 4.4 – Exemples d'écoulements fluides sur des corps solides de 2-D à 3-D

Dans un troisième cas, un fluide s'exerçant sur un solide et qui ne peut être modélisé comme

un fluide à deux dimensions ou axisymétrique, est dit fluide à trois dimensions, comme dans l'exemple de l'automobile présenté sur la figure 4.4(c).

4.2.4 Forces de « traînée » et de « portance »

4.2.4.1 Notions de « traînée » et de « portance »

Il est commun d'expérimenter la résistance que rencontre un corps quand il se déplace dans un fluide, comme par exemple lorsque nous nous déplaçons dans l'eau (résistance à un fluide liquide) ou quand nous portons un bras hors d'un véhicule en mouvement (résistance à un fluide à écoulement d'air).

Un fluide peut exercer des forces et des moments mécaniques sur un corps dans plusieurs directions. Il est donc très difficile de décrire précisément l'ensemble des phénomènes physiques qui agissent sur un corps. Pourtant, deux notions essentielles à la mécanique des fluides se dégagent de cette complexité :

- La « traînée » ou « force de traînée » : il s'agit de la force qu'un fluide exerce sur un corps dans la direction de l'écoulement du fluide.
- La « portance » ou « force de portance » : il s'agit de la force qu'un fluide exerce sur un corps dans la direction normale à l'écoulement du fluide.

A noter que pour bien comprendre les efforts exercés par un fluide sur un solide, il est nécessaire d'introduire la notion de « vitesse d'un fluide ». La vitesse d'un fluide approchant un corps est appelé « free-stream velocity » et prends généralement la notation V . Elle peut également s'écrire par u ou U . Quand le fluide est projeté sur l'axe (x), u est utilisée pour exprimer la composante en (x) de la vitesse.

La figure 4.5 représente les forces dues à un écoulement fluide de vitesse V et de direction horizontale, agissant sur un « corps à pointe » et résultant des forces de « traînée » et de « portance ». Nous noterons ces deux forces respectivement (F_D) et (F_L) pour garder la terminologie anglaise de Drag force (traînée) et Lift force (portance).

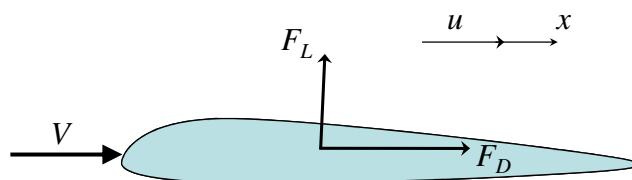


Fig. 4.5 – Forces agissant sur un « corps à pointe » et résultantes des forces de « traînée » et de « portance ».

Prenons le cas d'un fluide stationnaire. Ce dernier exerce seulement des forces de pression normales à la surface d'un corps immergé. Par contre, un fluide en mouvement exerce des forces tangentielles (*shear*) à la surface d'un corps provoquées par des conditions de « non-glissement » (no-slip) causée par les effets de viscosité. Ces deux forces ont, en général, des composantes dans

la direction du fluide et ainsi la force de traînée est due aux effets combinés de la pression et de surface mouillée (*wall shear*) dans la direction de l'écoulement fluide. Les composantes des forces de pression et de surface mouillée dans la direction normale à l'écoulement du fluide tendent à déplacer le corps dans cette direction, et leur somme est appelée « portance » ou « force de portance ».

Cas du fluide à trois dimensions

Pour décrire le cas d'un fluide à trois dimensions, nous rajouterons simplement une composante de force dans la direction normale du plan de manière à faire bouger le corps soumis à cet effort dans cette direction. Nous appellerons cette composante la « force de côté ».

Cas de la rotation

Les forces fluidiques peuvent aussi générer des moments mécaniques et causer la rotation d'un corps. Le moment appliqué à la direction du fluide est appelé le « rolling moment », celui appliqué à la direction de la portance le « yawing moment », et celui qui s'exerce sur la direction de la force de côté le « pitching moment ».

4.2.4.2 Calcul des forces de « traînée » et de « portance »

Les forces de pression et tangentielles agissant sur une surface élémentaire (dS) sont respectivement exprimées par $(P.dS)$ et $(\tau_w.dS)$, comme décrites sur la figure 4.6 dans le cas d'un corps à pointe.

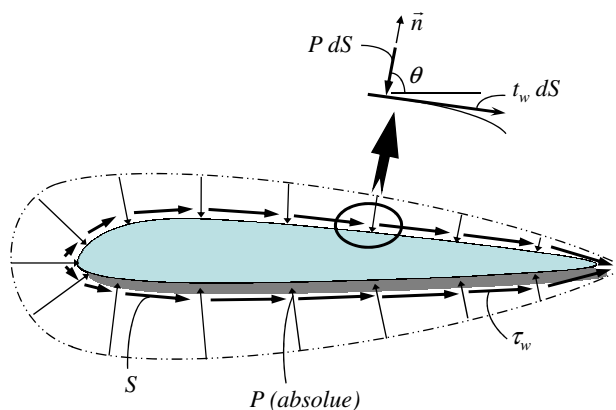


Fig. 4.6 – Description des forces de pression et tangentielles agissant sur une surface élémentaire dans le cas d'un corps à pointe.

Les forces de traînée et de portance élémentaires agissant sur (dS) dans un fluide à deux dimensions sont :

$$dF_D = -PdS\cos\theta + \tau_w dS\sin\theta \quad (4.2)$$

$$dF_L = -PdS\sin\theta + \tau_{\varpi}dS\cos\theta \quad (4.3)$$

où θ est l'angle que fait la normale à la surface élémentaire $dS(\vec{n})$ avec la direction positive de l'écoulement du fluide, comme l'illustre la figure 4.6.

Ainsi, les forces globales de traînée et de portance exercées sur le corps d'un solide sont déterminées par intégration des équations 4.2 et 4.3 sur toute la surface du corps.

$$\text{Force de traînée : } F_D = \int_S dF_D = \int_S (-P\cos\theta + \tau_{\varpi}\sin\theta)dS \quad (4.4)$$

et

$$\text{Force de portance : } F_L = \int_S dF_L = - \int_S (P\sin\theta + \tau_{\varpi}\cos\theta)dS \quad (4.5)$$

Les relations 4.4 et 4.5 sont souvent très utiles pour prédire les forces de traînée et de portance qui s'exercent sur le corps d'un solide quand le flux d'air est simulé à l'aide d'un ordinateur. Par contre, ces calculs s'avèrent peu pratiques lorsque l'on se trouve dans le cas de performances expérimentales. En effet, les détails de distribution des forces de pression et tangentielles sont très difficiles à déterminer par mesures directes. Fort heureusement, souvent ces informations ne sont pas nécessaires. Habituellement, tout ce dont nous avons besoin de savoir sont les forces résultantes de traînée et de portance agissant sur l'ensemble du corps solide, lesquelles peuvent être directement et facilement mesurées à partir d'une soufflerie.

Les équations 4.2 et 4.3 montrent qu'à la fois la surface mouillée et la pression, en général, contribuent à la traînée et à la portance.

4.2.4.3 Exemples d'application des forces de « traînée » et de « portance »

Cas d'une plaque sans épaisseur mise à l'horizontale

Dans le cas spécial d'une plaque sans épaisseur (ou négligeable), mise à l'horizontale, c'est-à-dire alignée parallèlement à la direction de l'écoulement d'air, la force de traînée dépend uniquement de la surface mouillée et est indépendant de la pression puisque $\theta = 90^\circ$. Ce cas est représenté sur la figure 4.7.

Cas d'une plaque sans épaisseur mise à la verticale

Quand la même plaque étudiée précédemment est placée normale à l'écoulement d'air, alors la force de traînée dépend uniquement de la pression et est indépendant de la surface mouillée. Cette dernière agit alors dans la direction normale à l'écoulement soit avec un angle $\theta = 0^\circ$, comme le montre la figure 4.8.

Finalement, si la plaque est inclinée avec un angle relatif à la direction de l'écoulement d'air, alors la force de traînée dans ce cas dépend à la fois de la pression et de la surface mouillée.

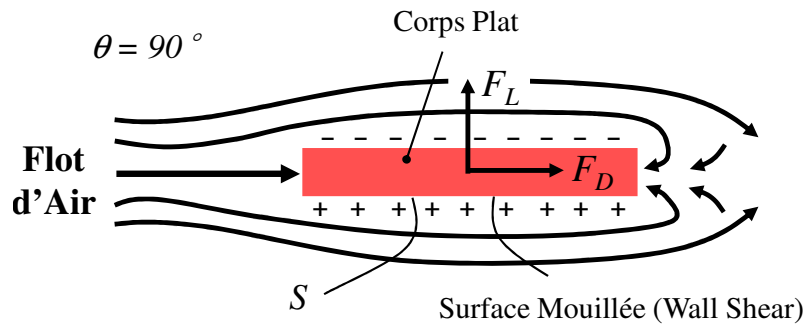


Fig. 4.7 – Traînée et portance agissant sur un corps plat et sans épaisseur parallèle à l'écoulement.

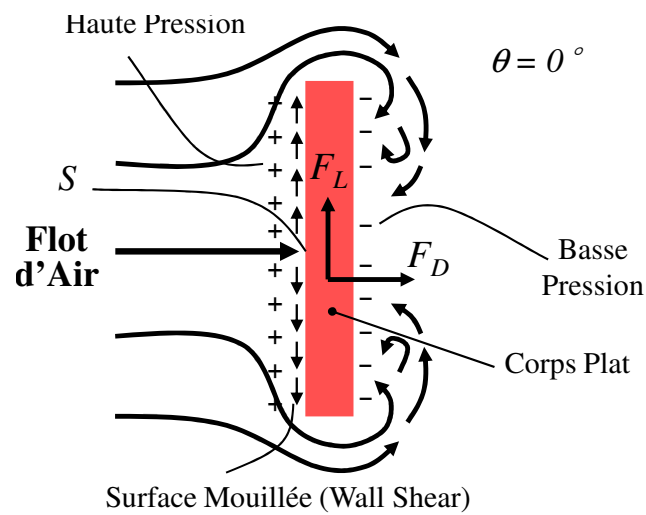


Fig. 4.8 – Traînée et portance agissant sur un corps plat et sans épaisseur normal à l'écoulement de l'air.

Cas d'un corps à pointe

Pour les « corps à pointe » comme les ailes des avions, la force tangentielle agit pratiquement parallèlement à la direction de l'écoulement d'air, et ainsi sa contribution à la portance est faible. La force de traînée pour de telles structures minces est principalement due à la force de frottement (*shear forces - the skin friction*). La figure 4.9 illustre cet exemple.

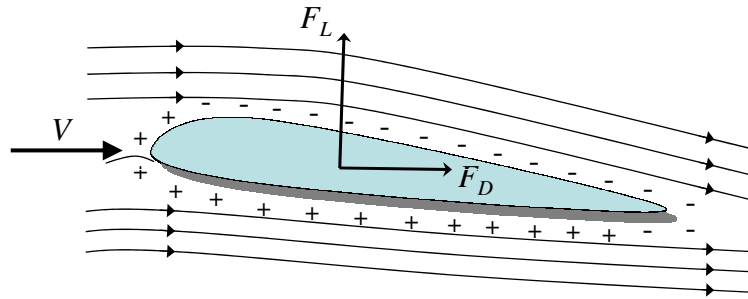


Fig. 4.9 – Forces de traînée et de portance agissant sur un corps à pointe.

En effet, les ailes des avions sont partagées et positionnées spécifiquement pour générer une portance avec un minimum de traînée. Ceci est réalisé en maintenant un angle d'attaque durant le vol. La traînée comme la portance sont fortement dépendante de cet angle d'attaque. La différence de pression entre le haut et le bas de la surface de l'aile génère une force de mouvement vers le haut qui tend à porter les ailes et ainsi l'avion auquel elles sont connectées. La figure 4.10 illustre la configuration d'écoulement sur aile d'avion et de la notion de profil associé.

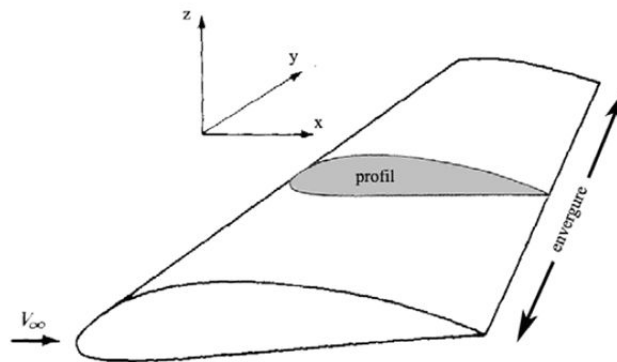


Fig. 4.10 – Illustration de la configuration d'écoulement sur aile d'avion et de la notion de profil associé.

4.2.4.4 Coefficients de « traînée » et de « portance »

Les forces de traînée et de portance dépendent entre autres de la densité (ρ) du fluide, de la vitesse amont V , et des dimensions du corps solide (taille, forme, orientation). Il n'est pas pratique de lister ces forces si l'on désire réaliser de nombreuses simulations. A la place, il est avantageux de travailler avec des données adaptées sans dimension qui représentent les

caractéristiques de la traînée et de la portance du corps solide. Ces données sont les coefficients de traînée (C_D) et de portance (C_L), lesquels sont calculés ainsi :

$$\text{Coefficient de traînée : } C_D = \frac{F_D}{\frac{1}{2}\rho V^2 S} \quad (4.6)$$

et

$$\text{Coefficient de portance : } C_L = \frac{F_L}{\frac{1}{2}\rho V^2 S} \quad (4.7)$$

Où (S) est ordinairement la surface frontale, qui est la surface projetée sur un plan normal à la direction de l'écoulement du corps solide. En d'autres mots, la surface frontale (S) est la surface qui pourrait être vue par une personne regardant vers le corps solide à partir de la direction du fluide approchant. Exemple : dans le cas d'un cylindre de diamètre (D) et de longueur (L), la surface frontale est de $S = LD$.

Dans le calcul du coefficient de portance, la valeur de la surface prise (S) est la surface de forme-plane, qui est la surface vue par une personne regardant vers le corps solide par-dessus à la normale au déplacement du fluide.

ρ est la densité du fluide (dans notre étude, c'est la densité de l'air). V est la vitesse du fluide appliquée sur le solide. A noter que le terme ($\frac{1}{2}\rho V^2$) que l'on trouve dans les équations 4.6 et 4.7 est appelé : « pression dynamique » (*dynamic pressure*).

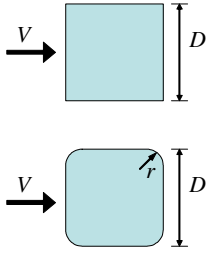
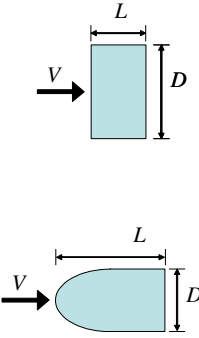
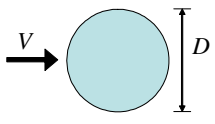
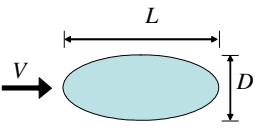
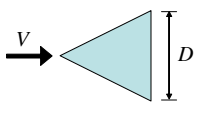
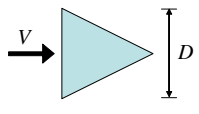
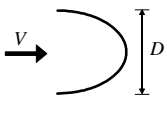
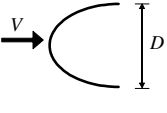
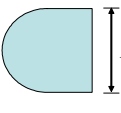
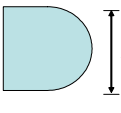
Pour déterminer numériquement les coefficients de traînée et de portance, nous pouvons nous baser sur de très nombreuses études (principalement expérimentales) dans le domaine de l'aérodynamisme. On trouve également une très grande quantité de données sur les coefficients de traînée dans la littérature pour à peu près toutes les formes de géométrie présentant un intérêt.

Les coefficients de traînée et de portance sont d'abord des fonctions de la forme du corps solide étudié. Cependant, dans certains cas ils sont aussi dépendants du nombre de Reynolds et de la rugosité de la surface. Les tableaux 4.2 et 4.3 donnent en exemple un certain nombre de coefficients de traînée en fonction de la forme à deux et trois dimensions du solide en question pour un nombre de Reynolds supérieur à 10^4 .

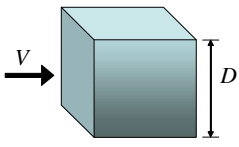
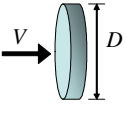
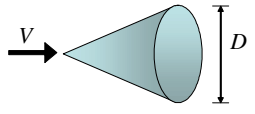
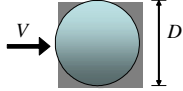
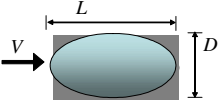
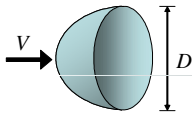
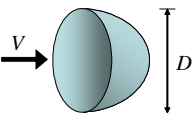
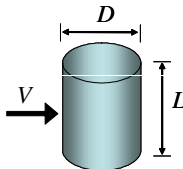
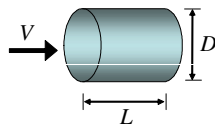
4.2.4.5 « traînée » de frottement et de pression (ou de forme)

Comme nous l'avons énoncé précédemment, la notion de force traînée est la force nette exercée par un fluide sur un corps solide dans la direction du flot due aux effets combinés des forces de pression et de surface mouillée. Il est souvent instructif de séparer les deux effets et de les étudier séparément, comme le montre la figure 4.7 précédemment :

- **traînée de frottement** : la part de traînée, qui est due directement au stress de la surface mouillée (τ_w), est appelée traînée de frottement (*friction drag*) et notée F_{D-frot} .

<p>Pièce Carrée</p>  <p><i>Coins Aigusés :</i> $C_D = 2.2$</p> <p><i>Coins Arrondis</i> ($r/D = 0.2$): $C_D = 1.2$</p>	<p>Pièce Rectangulaire</p>  <table border="1" data-bbox="1093 515 1220 716"> <thead> <tr> <th>L/D</th> <th>C_D</th> </tr> </thead> <tbody> <tr> <td>0.0*</td> <td>1.9</td> </tr> <tr> <td>0.1</td> <td>1.9</td> </tr> <tr> <td>0.5</td> <td>2.5</td> </tr> <tr> <td>1.0</td> <td>2.2</td> </tr> <tr> <td>2.0</td> <td>1.7</td> </tr> <tr> <td>3.0</td> <td>1.3</td> </tr> </tbody> </table> <p><i>Coins Aigusés :</i></p> <table border="1" data-bbox="1093 772 1220 929"> <thead> <tr> <th>L/D</th> <th>C_D</th> </tr> </thead> <tbody> <tr> <td>0.5</td> <td>1.2</td> </tr> <tr> <td>1.0</td> <td>0.9</td> </tr> <tr> <td>2.0</td> <td>0.7</td> </tr> <tr> <td>4.0</td> <td>0.7</td> </tr> </tbody> </table> <p><i>Bord de Front Arrondi :</i></p> <p><small>*Correspondant à une plaque fine</small></p>		L/D	C_D	0.0*	1.9	0.1	1.9	0.5	2.5	1.0	2.2	2.0	1.7	3.0	1.3	L/D	C_D	0.5	1.2	1.0	0.9	2.0	0.7	4.0	0.7
L/D	C_D																									
0.0*	1.9																									
0.1	1.9																									
0.5	2.5																									
1.0	2.2																									
2.0	1.7																									
3.0	1.3																									
L/D	C_D																									
0.5	1.2																									
1.0	0.9																									
2.0	0.7																									
4.0	0.7																									
<p>Pièce Arrondie (cylindre)</p>  <p><i>Écoulement Laminaire :</i> $C_D = 1.2$</p> <p><i>Écoulement Turbulent :</i> $C_D = 0.3$</p>	<p>Pièce Ellipticale</p>  <table border="1" data-bbox="1013 1019 1276 1198"> <thead> <tr> <th rowspan="2">L/D</th> <th colspan="2">C_D</th> </tr> <tr> <th>Laminaire</th> <th>Turbulent</th> </tr> </thead> <tbody> <tr> <td>2</td> <td>0.60</td> <td>0.20</td> </tr> <tr> <td>4</td> <td>0.35</td> <td>0.15</td> </tr> <tr> <td>8</td> <td>0.25</td> <td>0.10</td> </tr> </tbody> </table>		L/D	C_D		Laminaire	Turbulent	2	0.60	0.20	4	0.35	0.15	8	0.25	0.10										
L/D	C_D																									
	Laminaire	Turbulent																								
2	0.60	0.20																								
4	0.35	0.15																								
8	0.25	0.10																								
<p>Pièce Triangle Equilateral</p>  <p>$C_D = 1.5$</p>  <p>$C_D = 2.0$</p>	<p>Shell Semi-circulaire</p>  <p>$C_D = 2.3$</p>  <p>$C_D = 1.7$</p>	<p>Pièce Semi-circulaire</p>  <p>$C_D = 1.2$</p>  <p>$C_D = 1.7$</p>																								

Tab. 4.2 – Coefficients de traînée (C_D) de différents corps solides à deux dimensions (Reynould $Re > 10^4$ basé sur la surface frontal $S = b.D$ ou b est la longueur dans la direction normale au plan, appliquée lors d'une force de traînée de $F_D = C_D S \rho V^2 / 2$ où V est la vitesse amont).

<p>Cube, $A = D^2$</p>  <p>$C_D = 1.05$</p>	<p>Disque Fin Circulaire, $S = \pi D^2/4$</p>  <p>$C_D = 1.05$</p>	<p>Cône (pour $\theta = 30^\circ$), $S = \pi D^2/4$</p>  <p>$C_D = 0.5$</p>																										
<p>Sphère, $S = \pi D^2/4$</p>  <p>Laminar : $C_D = 0.5$ Turbulent : $C_D = 0.2$</p>	<p>Pièce Ellipticale, $S = \pi D^2/4$</p>  <table border="1" data-bbox="1037 862 1284 1064"> <thead> <tr> <th rowspan="2">L/D</th> <th colspan="2">C_D</th> </tr> <tr> <th>Laminaire</th> <th>Turbulent</th> </tr> </thead> <tbody> <tr> <td>0.75</td> <td>0.6</td> <td>0.2</td> </tr> <tr> <td>1</td> <td>0.5</td> <td>0.2</td> </tr> <tr> <td>2</td> <td>0.3</td> <td>0.1</td> </tr> <tr> <td>4</td> <td>0.3</td> <td>0.1</td> </tr> <tr> <td>8</td> <td>0.2</td> <td>0.1</td> </tr> </tbody> </table>		L/D	C_D		Laminaire	Turbulent	0.75	0.6	0.2	1	0.5	0.2	2	0.3	0.1	4	0.3	0.1	8	0.2	0.1						
L/D	C_D																											
	Laminaire	Turbulent																										
0.75	0.6	0.2																										
1	0.5	0.2																										
2	0.3	0.1																										
4	0.3	0.1																										
8	0.2	0.1																										
<p>Pièce Triangle Equilateral</p>  <p>$C_D = 0.4$</p>  <p>$C_D = 1.2$</p>	<p>Short Cylinder Vertical, $S = LD$</p>  <table border="1" data-bbox="790 1377 917 1612"> <thead> <tr> <th>L/D</th> <th>C_D</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>0.6</td> </tr> <tr> <td>2</td> <td>0.7</td> </tr> <tr> <td>5</td> <td>0.8</td> </tr> <tr> <td>10</td> <td>0.9</td> </tr> <tr> <td>40</td> <td>1.0</td> </tr> <tr> <td>∞</td> <td>1.2</td> </tr> </tbody> </table>	L/D	C_D	1	0.6	2	0.7	5	0.8	10	0.9	40	1.0	∞	1.2	<p>Short Cylinder Horizontal, $S = \pi D^2/4$</p>  <table border="1" data-bbox="1093 1344 1220 1534"> <thead> <tr> <th>L/D</th> <th>C_D</th> </tr> </thead> <tbody> <tr> <td>0.5</td> <td>1.1</td> </tr> <tr> <td>1</td> <td>0.9</td> </tr> <tr> <td>2</td> <td>0.9</td> </tr> <tr> <td>4</td> <td>0.9</td> </tr> <tr> <td>8</td> <td>1.0</td> </tr> </tbody> </table>	L/D	C_D	0.5	1.1	1	0.9	2	0.9	4	0.9	8	1.0
L/D	C_D																											
1	0.6																											
2	0.7																											
5	0.8																											
10	0.9																											
40	1.0																											
∞	1.2																											
L/D	C_D																											
0.5	1.1																											
1	0.9																											
2	0.9																											
4	0.9																											
8	1.0																											

Tab. 4.3 – Coefficients de traînée (C_D) de différents corps solides à trois dimensions (Reynould $Re > 10^4$ basé sur la surface frontale appliquée lors d'une force de traînée de $F_D = C_D S \rho V^2 / 2$ où V est la vitesse amont).

Elle est causée par les effets de frottement ou par les contraintes de cisaillement agissant tangentiuellement sur la surface du corps, comme l'illustre précédemment la figure 4.6.

La traînée de frottement est considérée comme nulle pour une surface plate normale à l'écoulement, et maximale pour une surface plate parallèle à l'écoulement. A noter aussi que la traînée de frottement est fortement fonction de la viscosité du fluide (plus le fluide est visqueux, plus la traînée de frottement est importante).

- **traînée de pression (ou forme)** : de même, la part qui est due directement à la pression (P), qui est engendrée par les forces de pression agissant perpendiculairement à la surface du corps, comme l'illustre précédemment la figure 4.6. Elle est appelée traînée de pression (*pressure drag*) ou traînée de forme (*form drag*), et notée F_{D-pres} , car elle dépend fortement de la forme du corps.

Si la traînée de pression est principalement proportionnelle à la surface frontale du corps immergé dans le fluide, elle l'est aussi à la différence entre les pressions agissant sur la face-avant et la face-arrière de ce dernier. Nous pouvons établir les coefficients de traînée de ces deux effets comme suit :

$$\text{Coefficient de traînée de frottement} : C_{D-frot} = \frac{F_{D-frot}}{\frac{1}{2}\rho V^2 S} \quad (4.8)$$

et

$$\text{Coefficient de traînée de pression (ou forme)} : C_{D-pres}L = \frac{F_{D-pres}}{\frac{1}{2}\rho V^2 S} \quad (4.9)$$

Où (S) est désignée comme la surface frontale ou la surface de forme-plane selon le type de traînée calculée. (ρ) est la densité du fluide (dans notre étude, c'est la densité de l'air). (V) est la vitesse du fluide appliqué sur le solide.

Quand les forces ou les coefficients de frottement et de pression sont disponibles, la force totale ou le coefficient global de traînée peut être déterminé par la simple addition des deux composantes :

$$C_D = C_{D-frot} + C_{D-pres} \quad (4.10)$$

$$F_D = F_{D-frot} + F_{D-pres} \quad (4.11)$$

4.3 Modèle physique de la matrice

4.3.1 Modèle en lévitation

4.3.1.1 Modèle de base

Le « modèle en lévitation » développé ici va décrire la mise en lévitation d'un objet par des jets d'air générés par la matrice de microactionneurs pneumatiques que nous étudions. Pour

élaborer un tel modèle, nous nous inspirerons de celui établi lors de l'établissement des forces de traînée et de portance agissant sur une « plaque sans épaisseur mise à la verticale ». En effet, l'objet manipulé peut être considéré comme une plaque sans épaisseur sur laquelle l'écoulement d'air agit de façon normale. Seule l'orientation du corps et de l'écoulement ont changé. Ainsi, une simple rotation de 90° du modèle de base de la « plaque sans épaisseur mise à la verticale » suffit à représenter l'objet à l'horizontale en maintenant un écoulement normal de l'air sur le solide, comme le montre la figure 4.11(a). La matrice sera alors représentée par une rangée d'orifices, qui sera rajoutée au modèle précédent, pour représenter les jets d'air s'exerçant de manière normale sur l'objet, comme le montre la figure 4.11(b).

Nous retrouverons ainsi une configuration fluide assez similaire à la précédente dans le cas du corps à la verticale. Par contre, nous verrons que le rôle des forces de traînée et de portance s'est inversé. En effet, ici la force de traînée (force qui s'exerce sur un corps dans le sens de la direction de l'écoulement) est appliquée pour maintenir le corps en lévitation stationnaire et non pour créer un mouvement comme précédemment. Nous appellerons cette force : force de lévitation en la notant (F_L).

Par conséquent, les seules relations du modèle statique concerneront la force de lévitation, laquelle dépend des principes de la traînée étudiés précédemment. Ainsi, nous définirons deux composantes de la force de lévitation :

- La composante de pression de la force de lévitation, notée (F_{L-pres}), qui se base sur les principes de calcul du coefficient de pression, noté (C_{xp}). Il s'agit de l'effort qu'exerce l'écoulement d'air sur l'objet lorsqu'il est normal à la surface de ce dernier.
- La composante de frottement, notée (F_{L-frot}) qui se base sur les principes de calcul du coefficient de frottement, noté (C_{xf}). Il s'agit de l'effort qu'exerce l'écoulement d'air sur l'objet lorsqu'il frotte à la surface de ce dernier.

Les relations 4.12, 4.13 et 4.14 font état de cette décomposition de la force de lévitation :

$$F_L = F_{L-pres} + F_{L-frot} \quad (4.12)$$

$$F_{L-pres} = \frac{1}{2}\rho C_{xp} V_a^2 S \quad (4.13)$$

$$F_{L-frot} = \frac{1}{2}\rho C_{xf} V_a^2 S \quad (4.14)$$

avec (ρ) la densité du fluide, (v_a) la vitesse en amont du système, (S) la surface de l'objet normale à l'écoulement du fluide, et (C_{xp}, C_{xf}) respectivement les coefficients de pression et de frottement qui s'exercent sur l'objet normale à l'écoulement d'air.

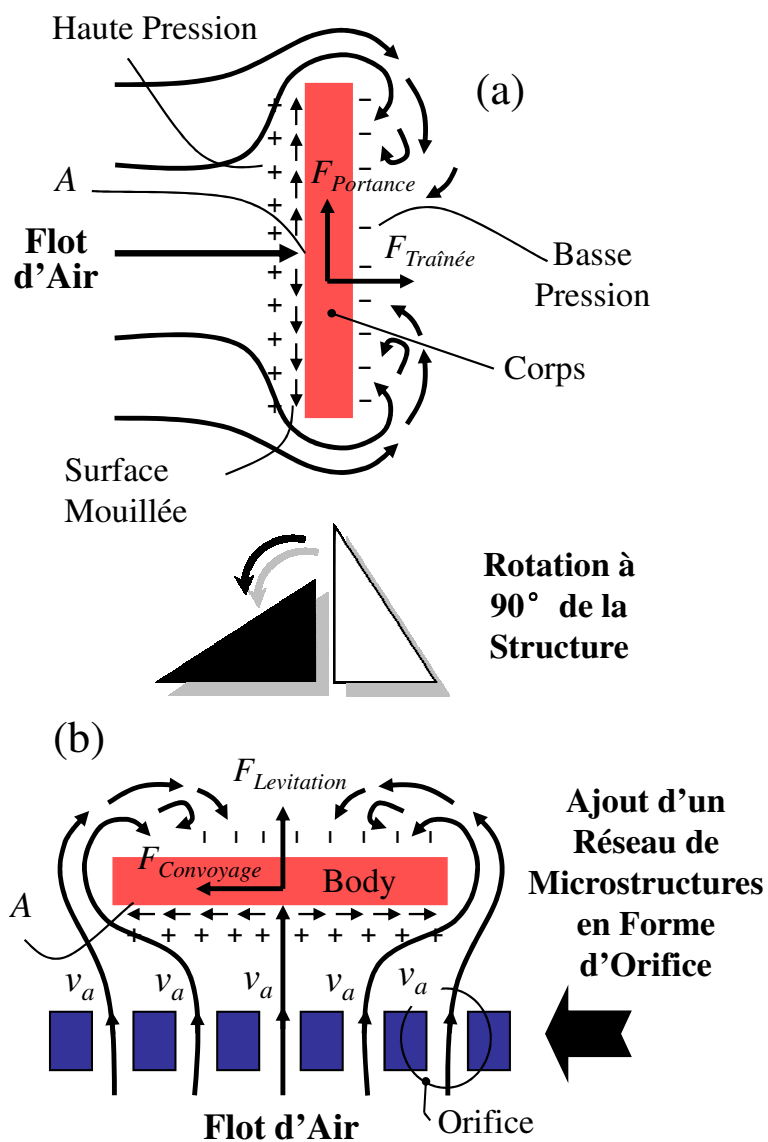


Fig. 4.11 – Approche de modélisation de la matrice de microactionneurs pneumatiques.

4.3.2 Modèle de convoyage en lévitation

4.3.2.1 Description globale

Par le modèle de convoyage en lévitation, nous décrivons le déplacement d'un objet à partir d'un écoulement d'air approprié. Nous définirons ainsi les interactions fluidiques entre la matrice de microactionneurs pneumatiques, constituant la surface active du dispositif de micromanipulation, et l'objet manipulé lui-même. Plus précisément, nous établirons les relations propres aux forces de lévitation (F_L) et de convoyage (F_C) de l'objet lors d'une séquence de manipulation.

La figure 4.12 illustre ce modèle, en détaillant le dispositif, l'objet et l'ensemble des forces et effets fluidiques qui s'exercent sur ce dernier. Nous reviendrons dans la suite sur plusieurs des éléments exprimés sur cette figure.

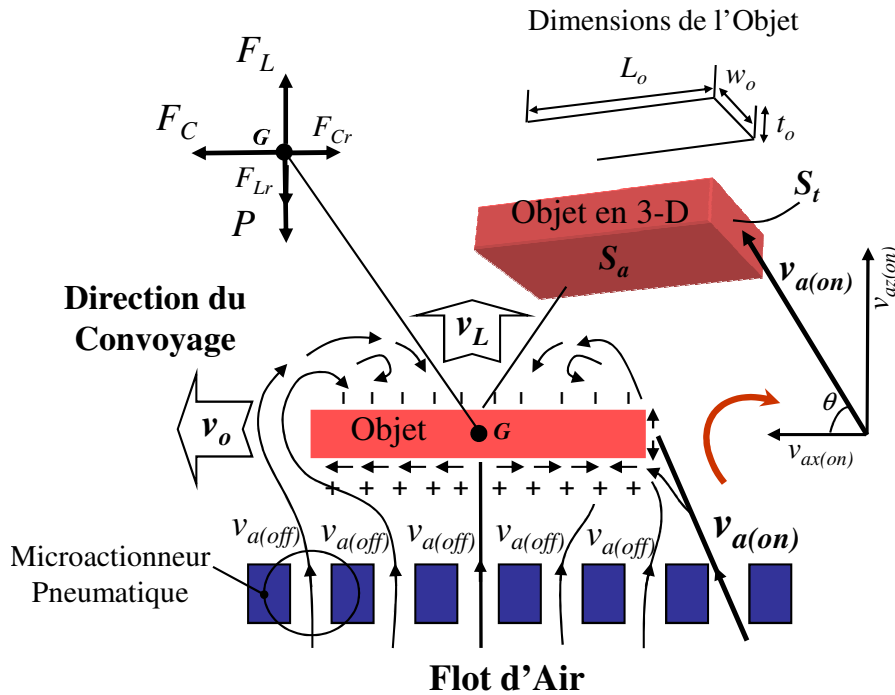


Fig. 4.12 – *Modèle en lévitation et déplacement de la matrice.*

Dans le « modèle de convoyage en lévitation », les microactionneurs pneumatiques sont représentés par la même rangée de microstructures en forme d'orifice adoptée dans le « modèle en lévitation ». Par contre, ils prennent à présent deux positions fonctionnelles :

- Position au repos : la vitesse du flot d'air en sortie de l'orifice est de direction verticale et d'amplitude $v_{a(off)}$ (Cf. figure 4.12).
- Position active : la vitesse du flot d'air en sortie de l'orifice peut prendre deux orientations inclinées (angle d'inclinaison : θ) suivant qu'on l'active à droite ou à gauche du plan de mouvement. Son amplitude change également et aura pour valeur : $v_{a(on)}$ (Cf. figure 4.12).

Dans le reste de notre étude, nous définirons les deux vitesses introduites précédemment à

partir de leur composant dans le plan de mouvement de l'objet (x, z) :

$$v_{a(off)} = v_{a(off)} \quad (\text{pas de composante en } x) \quad (4.15)$$

$$v_{a(on)} = v_{ax(on)} + v_{az(on)} \quad (4.16)$$

4.3.2.2 Forces fluidiques

Chacune des forces fluidiques du modèle dynamique s'applique au point de gravité de l'objet (G), comme le décrit la figure 4.12. Par l'application combinée des flots d'air de vitesse $v_{a(off)}$, et $v_{a(on)}$, des forces de lévitation (F_L) et de convoyage (F_C) vont s'exercer sur l'objet, l'amenant à se déplacer en lévitation vers la droite ou vers la gauche selon l'orientation du flot d'air de vitesse $v_{a(on)}$.

Dans ce qui suit, nous allons détailler comment se composent les deux principales forces fluidiques :

- Les forces de « lévitation » : les forces doivent permettre à l'objet de rester en lévitation, à l'arrêt comme lors du déplacement de ce dernier. Elles sont regroupées en une seule force nommée : force de lévitation (F_L), laquelle est la combinaison de deux composantes de force fluidique :
 - la composante de force de pression (F_{L-pres}) exercée par les flots d'air verticaux de vitesse $v_{a(off)}$ en sortie des micro-valves,
 - la composante de force de frottement (F_{L-frot}) exercée par le flot d'air, de vitesse $v_{a(on)}$, orienté sur la tranche arrière de l'objet en déplacement.
- Les forces de « convoyage » : les forces doivent permettre à l'objet de se déplacer vers la droite ou vers la gauche du plan de représentation. Elles sont regroupées en une seule force nommée : force de convoyage (F_C), laquelle est la combinaison de deux composantes de force fluidique :
 - la composante de force de pression (F_{C-pres}) exercée par le flot d'air, de vitesse $v_{a(on)}$, orienté sur la tranche arrière de l'objet en déplacement.
 - la composante de force de frottement (F_{C-frot}) exercée par les flots d'air verticaux de vitesse $v_{a(off)}$ en sorties des micro-valves,

On supposera dans un premier temps que les composantes liées au frottement (F_{L-frot}) et (F_{C-frot}) comme négligeables. En effet, les effets de la pression sont largement prépondérants sur les frottements. Le peu de force de pression qui intervient dans la traînée a donc une grosse influence sur la valeur de celle-ci. Par conséquent, dans la suite de notre étude, nous considérerons toujours les composantes liées au frottement comme nulles en présence de force de pression :

$$F_{L-frot} \approx 0 \quad (4.17)$$

$$F_{C-frot} \approx 0 \quad (4.18)$$

Finalement, la relation de la force de lévitation peut s'établir de la manière suivante :

$$F_L = F_{L-frot} + F_{L-pres} \cong F_{L-pres} \quad (4.19)$$

$$F_{L-pres} = \frac{1}{2} \rho C_{xp} v_{az(off)}^2 S_a \quad (4.20)$$

avec la grandeur (S_a) représentant la dimension de la surface exposée par les flots d'air de la composante en (z) de la vitesse $v_{a(off)}$, soit ($v_{az(off)}$), (ρ) la densité du fluide et (C_{xp}, C_{xf}) respectivement les coefficients de « traînée » de pression et de frottement qui s'exercent sur l'objet normal à l'écoulement d'air.

De la même façon, nous établirons la relation de la force de convoyage de la manière suivante :

$$F_C = F_{C-frot} + F_{C-pres} \cong F_{C-pres} \quad (4.21)$$

$$F_{C-pres} = \frac{1}{2} \rho C_{xp} v_{ax(on)}^2 S_t \quad (4.22)$$

avec la grandeur (S_t) représentant la dimension de la surface exposée par le flot d'air de la composante en (x) de la vitesse $v_{a(on)}$, soit ($v_{ax(on)}$), (ρ) la densité du fluide et (C_{xp}, C_{xf}) respectivement les coefficients de « traînée » de pression et de frottement qui s'exercent sur l'objet normal à l'écoulement d'air.

4.3.2.3 Forces résiduelles et de résistance

Forces résiduelles

On a vu précédemment qu'en changeant l'angle que fait le corps avec l'écoulement d'air, on modifiait la portance et la traînée, pourquoi ? En fait, la forme de l'écoulement autour du profil varie avec l'incidence de ce dernier. Elle modifie donc la répartition des pressions sur la surface du corps solide :

- à incidence nulle : la portance est quasiment nulle (les filets d'air sont peu déviés, les surpressions et dépressions sont faibles), la traînée est faible,
- à moyenne incidence : la déviation des filets d'air crée des surpressions et dépressions, la portance augmente ainsi que la traînée,
- à incidence plus forte : des tourbillons apparaissent à l'arrière de l'extrados. La viscosité de l'air fait que les filets d'air n'ont plus assez d'énergie pour rester collés au profil, ils décrochent en formant des tourbillons qui ont une position aléatoire. Ceux-ci génèrent, à un instant donné, une surpression sur la surface du corps, puis à l'instant suivant une dépression au même endroit. La portance varie dans le temps, le profil est *décroché* et ne peut plus assurer sa sustentation (équilibre, poids/portance).

Comme on peut l'entrevoir, il est très difficile, pour ne pas dire quasi impossible, de modéliser correctement l'ensemble de ces forces résiduelles. Ainsi, pour des raisons de simplification de notre modèle, nous ne prendrons pas en compte ces phénomènes.

Forces de résistance

Lorsque l'objet est en lévitation ou en convoyage, il s'exerce sur lui des forces de résistance qui s'opposent à son mouvement. Nous distinguerons deux forces de résistance :

- La force de résistance à la lévitation (F_{Lr}) : cette force de résistance se produit de haut en bas lorsque l'objet est placé en lévitation avec une vitesse (v_L).
- La force de résistance au convoyage (F_{Cr}) : cette force de résistance se produit dans la direction du convoyage de l'objet se déplaçant à une vitesse de (v_o).

Le calcul de ces forces est donné par les relations suivantes :

$$F_{Lr} = \frac{1}{2} \rho C_{xp} v_L^2 S_a \quad (4.23)$$

$$F_{Cr} = \frac{1}{2} \rho C_{xp} v_o^2 S_t \quad (4.24)$$

avec les mêmes grandeurs utilisées dans les relations 4.20 et 4.22.

4.3.3 Modèle dynamique

4.3.3.1 Modèle à une dimension de convoyage

Pour élaborer le modèle dynamique de la matrice, nous appliquons, sur les axes (x) et (z), la relation de la dynamique que les forces fluidiques et mécaniques exercent sur l'objet.

Ainsi, la projection sur l'axe horizontal (x) de la relation de la dynamique s'écrit :

$$F_C - F_{Cr} = m \frac{d^2 x}{dt^2} \quad (4.25)$$

avec (m) la masse de l'objet, (F_{Cr}) la force de résistance de convoyage dont la relation est donnée en 4.24 et (F_C) la force de convoyage que nous pouvons calculer à partir de la relation 4.21.

De même, sur l'axe vertical (z) la relation de la dynamique s'écrit :

$$F_L - F_{Lr} - P = m \frac{d^2 z}{dt^2} \quad (4.26)$$

avec (m) la masse de l'objet, (F_{Lr}) la force de résistance de lévitation dont la relation est donnée en 4.23, (F_L) la force de lévitation que nous pouvons calculer à partir de la relation 4.19, et (P) le poids de l'objet : $P = mg$ (g : coefficient de gravité).

4.3.3.2 Modèle à deux dimensions de convoyage

Pour obtenir le modèle à deux dimensions de convoyage de la matrice, nous établirons la relation de la dynamique sur les trois axes (x), (y) et (z). Nous reprendrons pour cela les

relations 4.25 et 4.26 et ajouterons la projection sur l'axe horizontal (y), tel que :

$$F_C - F_{Cr} = m \frac{d^2 y}{dt^2} \quad (4.27)$$

4.4 Description VHDL-AMS

4.4.1 Arborescence

Aborder la description du modèle comportemental de manière globale, sans étape intermédiaire, induit inévitablement des erreurs et des difficultés de développement. Dans ce travail de description en VHDL-AMS, nous suivrons une méthode de conception basée sur une arborescence simple et progressive du modèle visé. Cette approche nous permettra de garder un héritage de chaque phase de description validée.

Dans la description du modèle comportemental de la matrice, l'arborescence que nous avons construite est composée de cinq étapes de modélisation correspondant à trois niveaux de description comportementale :

- Niveau 1 : Lévitacion \longrightarrow Axe (z)
Étapes 1 et 2 : Cas simple et cas particulier
- Niveau 2 : Convoyage à une dimension (1-D) en lévitation \longrightarrow Axe (x, z)
Étape 3 : Cas en boucle ouverte
- Niveau 3 : Convoyage à deux dimensions (2-D) en lévitation \longrightarrow Axe (x, y, z)
Étapes 4 et 5 : Cas en boucle ouverte et boucle fermée

La figure 4.13 donne un aperçu de l'arborescence que nous utiliserons tout au long du développement du modèle comportemental de la « matrice ».

4.4.2 Structure Générale

4.4.2.1 Approche par unités de conception

Comme nous l'avons introduit dans le chapitre 3, la structure générale du programme VHDL-AMS se base sur cinq types d'unité de conception (*design unit*).

Dans la construction de notre modèle comportementale de la « matrice » en VHDL-AMS, nous définirons d'abord les unités de conception du modèle global. La figure 4.14 illustre la structure générale du programme VHDL-AMS que nous adopterons et que nous détaillerons par la suite.

4.4.2.2 Détails des unités de conception

Déclaration d'entité

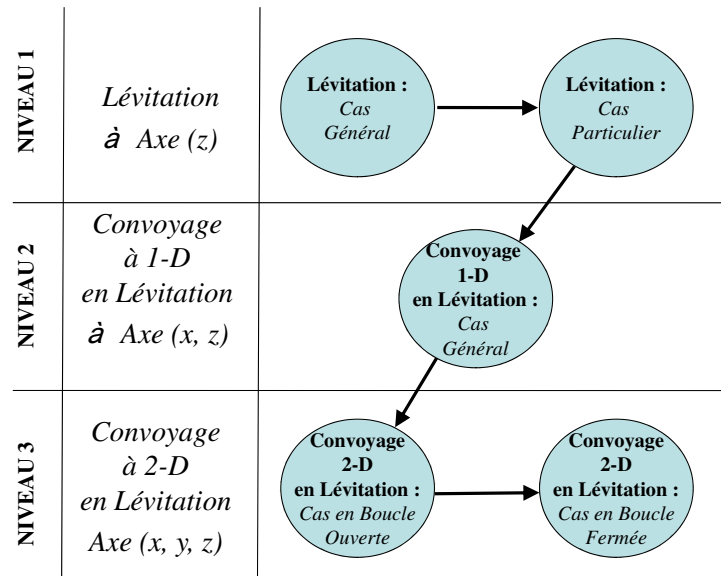


Fig. 4.13 – Modèle de detection distribuée de l'objet sur la « matrice ».

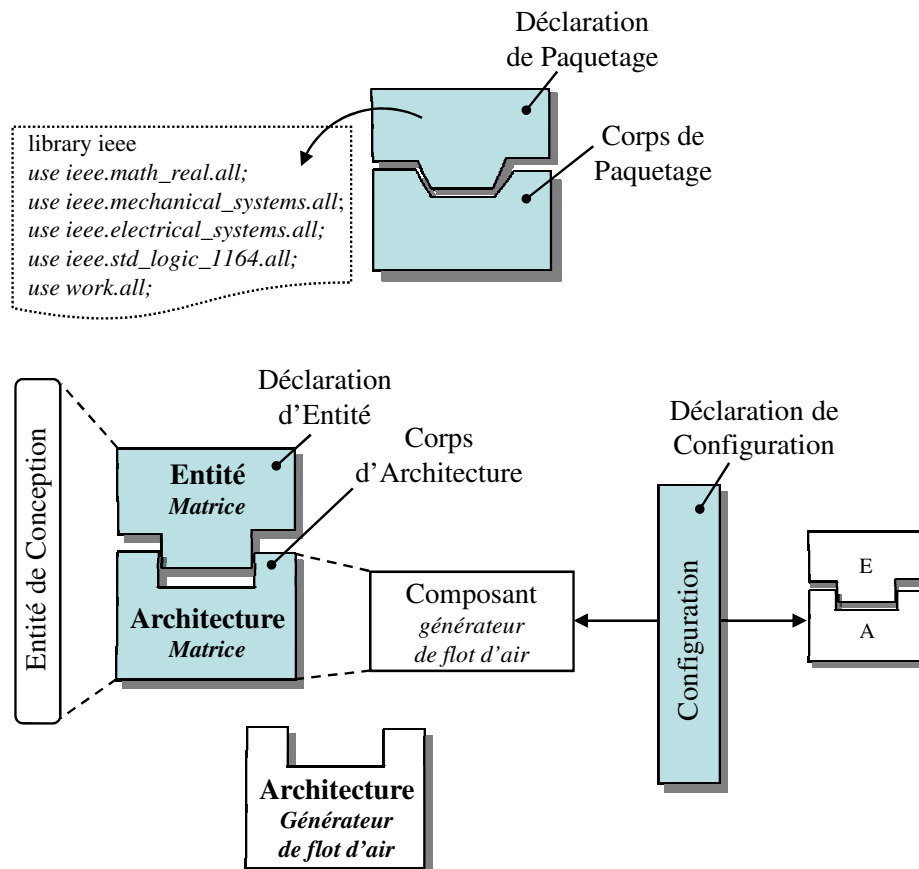


Fig. 4.14 – Structure générale du modèle VHDL-AMS de la « matrice ».

L'entité de la conception, que nous nommerons « matrice », est présentée sur la figure 4.15. Nous montrons ainsi que l'interface du modèle de la « matrice » n'est constituée que de paramètres génériques et de ports de classe terminal. Les paramètres génériques serviront à rendre le modèle plus général, alors que les ports de classe terminal définiront les points de connexions à temps continu non directionnels pour lesquels les lois de Kirchhoff sont satisfaites.

```

ENTITY Matrice is
  GENERIC(
    Mo: real:=0.01;    -- Masse de l'objet (kg)
    R: real:= 0.05;    -- Rayon de l'objet
    Vaoff: real:= 3.0; -- Velocité du flot d'air (Valve OFF)
    Tho: real;        -- Epaisseur de l'objet
    Dens: real:=1.3;  -- Densité de l'air
    Cxf: real:= 0.004; -- Coefficient de traînée de frottement
    Cxp: real:= 1.11; -- Coefficient de traînée de pression
    DD: real;        -- Distance entre le bord de l'objet
    -- et l'ouverture de la micro-valve
    DH: real;        -- largeur de l'ouverture de la micro-valve
    DIM: integer;    -- dimension de la surface distribuée
    Ds0, Vox0, Height0: real -- Valeurs initiales
  );
  PORT(terminal FA,FV:translational);
END Entity Matrice;

```

Fig. 4.15 – Déclaration de l'entité de la cellule de base de la « matrice »

Corps d'architecture

Le modèle comportemental VHDL-AMS de la « matrice » doit être une description comportementale et globale au plus haut niveau d'abstraction du système. Par conséquent, il doit être le plus compact possible et tenir si possible dans une seule architecture du corps d'architecture. Cependant, pour faciliter le test de notre modèle nous adopterons une structure globale intégrant un composant de conception pour la représentation comportementale du « générateur de flot d'air ».

Le détail de l'architecture de ce composant est donné dans l'Annexe. De plus, la principale partie du corps d'architecture de cette conception est détaillée dans la partie suivante.

Déclaration et corps de paquetage

Les déclarations de paquetage se basent essentiellement sur la bibliothèque *ieee (library ieee)*, qui regroupe les paquetages :

- *ieee.math_real.all* (paquetage regroupant des grandeurs et fonctions mathématiques au format « réel »);
- *ieee.std_logic_1164.all* (paquetage regroupant des grandeurs au format « logique »);
- *ieee.mechanical_systems.all* (paquetage regroupant des grandeurs physiques de natures

- « mécaniques »);
- `ieee.electrical_systems.all` (paquetage regroupant des grandeurs physiques de natures « électriques »).

4.4.3 Description du modèle « Lévitación : Cas Général »

4.4.3.1 Éléments de modélisation

La première phase de description concerne la modélisation comportementale de l'objet maintenu en lévitation par la matrice de microactionneurs pneumatiques. A ce niveau de description, nous donnons les hypothèses comportementales de la matrice et de l'objet permettant à ce dernier de se placer en lévitation sur l'axe (z), comme le montre la figure 4.16 :

- L'objet est plat, rectangulaire et homogène ;
- Un seul flot d'air est appliqué sous l'objet ;
- La direction du flot d'air est verticale et invariable ;
- Le flot d'air couvre la totalité de la surface mouillée de l'objet (surface en face-arrière) ;
- L'application de la force est homogène sur toute la surface mouillée de l'objet ;
- La force de lévitation s'applique sur le centre de gravité de l'objet.

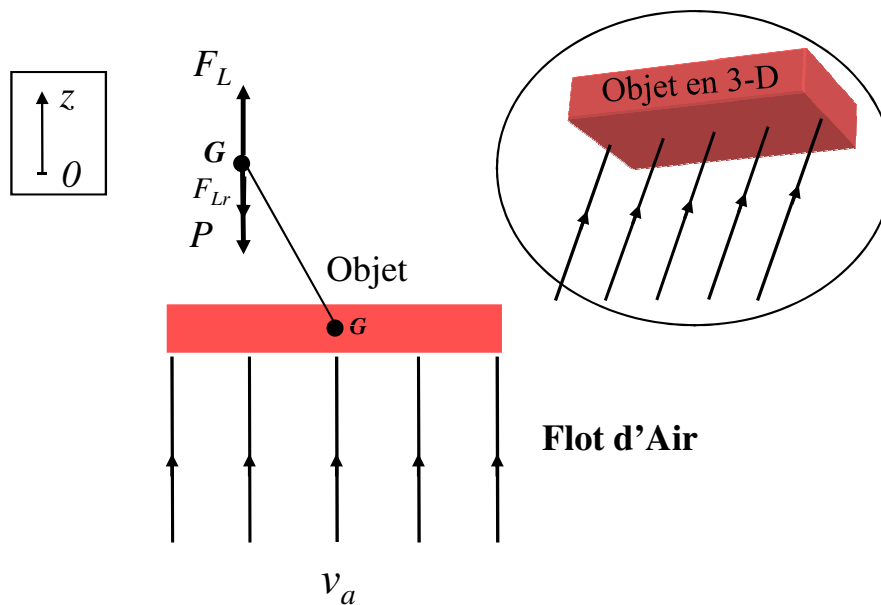


Fig. 4.16 – Modèle de l'objet en lévitation.

4.4.3.2 Éléments de programmation

Les principales étapes de l'écriture du modèle VHDL-AMS « lévitation : cas général » sont représentés sur la figure 4.17. Après les déclarations des objets (*constant*, *quantity* et *signal*), la description reprend le modèle physique de la « matrice » établi précédemment à partir de

simples relations de calcul.

```

ARCHTECHTURE behavior Matrice is

    constant g : real := 9.8;      -- Gravitation (en m/s2)
    constant S : real ;          -- Surface où la force de lévitation s'applique
    quantity Dz : real;          -- Position de l'objet sur l'axe (z)
                                -- (hauteur de l'objet)

    quantity Voz : real;         -- Vélocité de l'objet sur l'axe (z)
    quantity FL : real;         -- Force de lévitation
    quantity FLr : real;        -- Force de résistance en lévitation
    quantity P : real;          -- Poids de l'objet
    quantity Vaoff across FV ;  -- Vélocité de l'air (micro-valve au repos)

    signal Voz_s : real;
    signal fall : boolean;      -- choc de l'objet avec le sol

    BEGIN

    -- Initiation
    break Dz => Dz0, Voz => Voz0;

    -- Calculs des forces
    P == Mo*g;
    FL == 0.5*Dens*Cxp*Vazoff**2*S;

    -- Rebond de l'objet quand il heurt le sol
    break Voz => - Voz when fall;
    fall <= not Dz'above(0.0);
    break on fall;

    -- Calculs des forces de resistance
    Voz_s <= Voz'above(0.0);
    break on Voz_s;
    if Voz_s use
        FLr == 0.5*Dens*R**2*Cxp*Voz**2;
    else
        FLr == - 0.5*Dens*R**2*Cxp*Voz**2;
    end use;

    -- Relations de la dynamique
    FL - P - FLr == Mass*Voz'dot;
    Voz == Dz'dot;

    END;

```

Fig. 4.17 – Programmation VHDL-AMS du modèle « lévitation : cas général ».

Comme on peut le voir, le langage VHDL-AMS est surtout marqué par sa simplicité d'écriture, reprenant souvent directement les relations établies théoriquement.

Il permet également, via le développement de certaines spécificités de langage, de modéliser rapidement des comportements souvent complexes à simuler. C'est ce que nous verrons dans les parties de notre modèle que nous avons intitulé : « rebond de l'objet quand il heurte le sol » et « force de résistance en lévitation ». En effet, avec l'instruction « break », nous pouvons facilement gérer une discontinuité des valeurs initiales et de la solution des équations différentielles codées par des instructions simultanées.

4.4.4 Programme du modèle « Convoyage à 1-D en lévitation »

4.4.4.1 Éléments de modélisation

Le précédent modèle « lévitation » sera considéré, par la suite, comme la cellule de base de notre modèle de matrice de microactionneurs pneumatiques. Nous reprendrons donc cet élément dans la construction du programme pour le modèle « convoyage à 1-D en lévitation ».

Cellule de base

Pour permettre le déplacement de l'objet sur l'axe (x) tout en le maintenant en lévitation, nous allons donner à notre cellule de base de nouvelles propriétés. Nous nous inspirerons pour cela des fonctions de « micro-valve bi-directionnelle » du microactionneur pneumatique, comme les décrivent les figures 4.18(a) et 4.18(b). En intégrant ces nouveaux éléments, notre nouvelle cellule de base peut alors se décrire ainsi :

- Le modèle de la cellule est de forme carrée ou rectangulaire pour s'approcher de la cellule réelle.
- Le modèle de la cellule possède un orifice de forme rectangulaire allongée en son milieu.
- Le modèle de la cellule possède une micro-valve mobile qui se trouve alignée sous l'orifice permettant à la cellule basée sur le microactionneur pneumatique de prendre deux états fonctionnels :
 - Un état repos : Micro-valve non actionnée générant un flux d'air vertical, comme le montre la figure 4.18(a) ;
 - Un état actif : Micro-valve actionnée et se déplaçant à droite ou à gauche selon la commande et générant un flux d'air incliné selon un angle (θ), comme le montre la figure 4.18(b).

Dispositif de cellules alignées

Une fois la cellule de base du microactionneur pneumatique définie dans son environnement fluide, nous alignerons ces éléments de manière à en constituer une rangée et élaborer un dispositif de convoyage à une dimension (1-D) en lévitation sur les axes (x) et (z). La figure 4.19 illustre cette approche en montrant les vues de dessus et en section du système.

Le modèle représenté se compose de cinq cellules « microactionneurs pneumatiques », alignées côte à côte de manière rectiligne. Les quatre premières cellules en partant de la gauche sont à l'état repos, alors que la cinquième cellule est à l'état actif (génération d'un flux d'air orientée à gauche).

Complément du modèle

Nous compléterons notre nouveau modèle en listant un certain nombre d'éléments complémentaires de modélisation, comme suit :

- L'objet est plat, carré (ou rectangulaire) et homogène ;
- Le flot d'air n'est pas compressible ;
- Le flux d'air ne perd pas de sa vitesse en sortant de l'orifice de la cellule ;

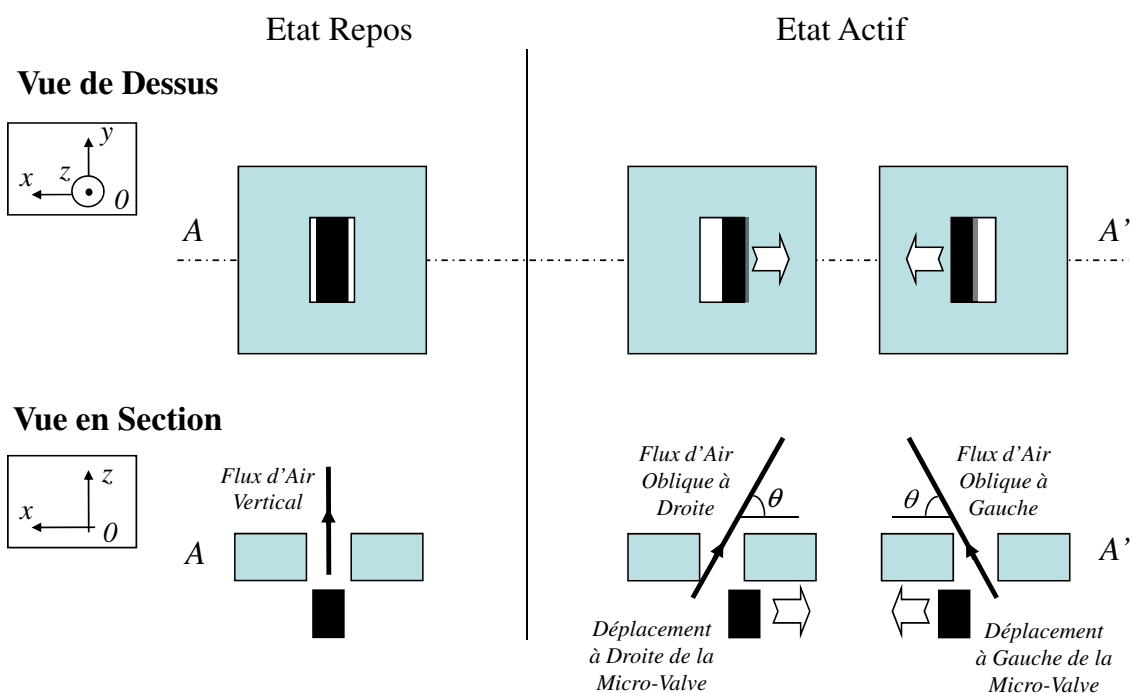


Fig. 4.18 – Modélisation de la cellule « microactionneur pneumatique ».

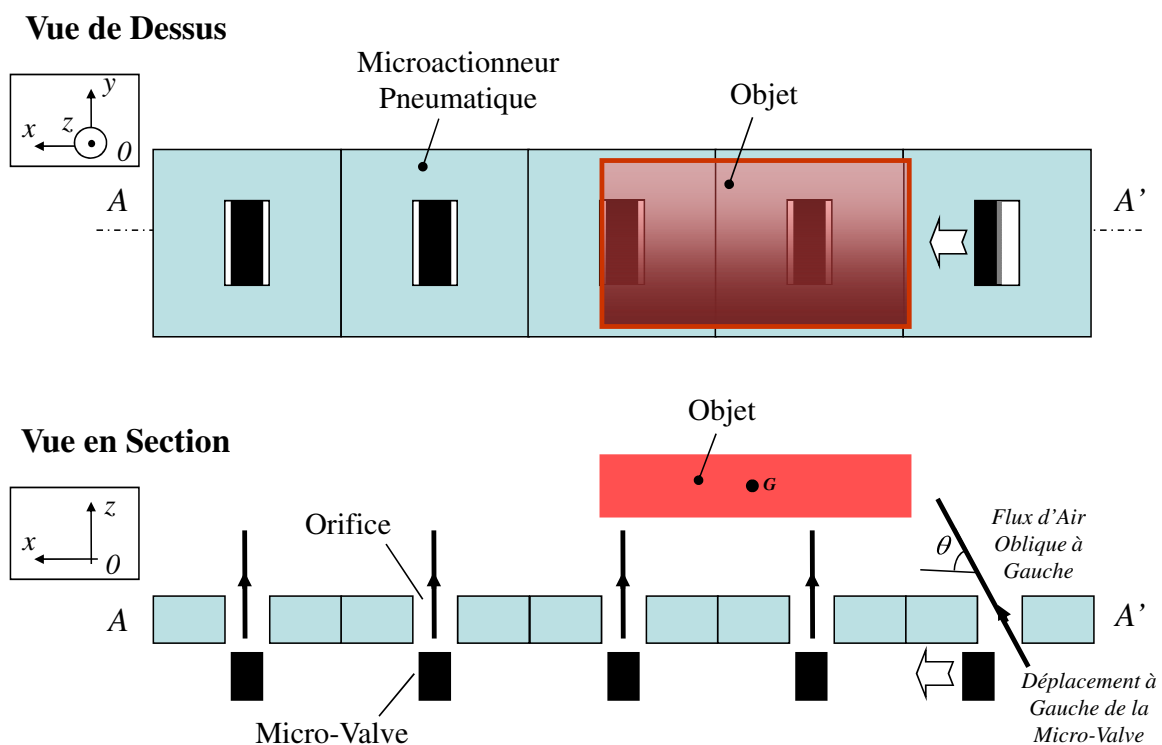


Fig. 4.19 – Description de cellules « microactionneurs pneumatique » pour convoyage 1-D.

- La vitesse du flux d'air est indépendante de la hauteur de la cellule ;
- Seule la cellule qui détecte l'extrémité arrière (dans le sens du mouvement) de l'objet est activée ;
- L'inclinaison de l'objet lors de son convoyage en lévitation est négligée ;
- L'application de la force est homogène sur l'ensemble de la surface mouillée de l'objet ;
- Toutes les forces s'appliquent au centre de gravité (G) de l'objet ;
- Le dispositif des cellules alignées est supposé sans bord ;
- L'objet ne se déplace pas sur l'axe (y).

4.4.4.2 Éléments de programmation

Déclaration d'entité

La déclaration d'entité du modèle « convoyage à 1-D en lévitation » se base sur la déclaration du modèle « lévitation : cas général » à laquelle on a ajouté les nouveaux éléments du programme (en gras), comme le montre la figure 4.20.

ARCHTECHTURE behavior <i>Matrice</i> is	
<i>constant g : real := 9.8;</i>	-- Gravitation (en m/s2)
● <i>quantity Dx : real;</i>	-- Position de l'objet sur l'axe x
<i>quantity Dz : real;</i>	-- Position de l'objet sur l'axe z (hauteur de l'objet)
● <i>quantity Vox : real;</i>	-- Vitesse de l'objet sur l'axe x
<i>quantity Voz : real;</i>	-- Vitesse de l'objet sur l'axe z
<i>quantity FL : real;</i>	-- Force de lévitation
<i>quantity FLr : real;</i>	-- Force de résistance en lévitation
● <i>quantity FC : real;</i>	-- Force de convoyage
● <i>quantity FCr : real;</i>	-- Force de résistance en convoyage
<i>quantity P : real;</i>	-- Poids de l'objet
● <i>quantity Vax-on : real :</i>	-- Vitesse du flot d'air sur l'axe x (micro-valve activée)
● <i>quantity Vaz-on : real :</i>	-- Vitesse du flot d'air sur l'axe z (micro-valve activée)
● <i>quantity Sb1,Sb2,Sb3 : real;</i>	-- Surface partielle exposée de l'objet
● <i>quantity Sb : real;</i>	-- Surface totale exposée de l'objet
● <i>quantity position_left : real;</i>	-- Position de l'extrémité gauche de l'objet
● <i>quantity position_right : real;</i>	-- Position de l'extrémité droite de l'objet
<i>quantity Va-off across FV ;</i>	-- Vitesse du flot d'air (micro-valve non activée)
● <i>quantity Theta across FA ;</i>	-- Angle d'orientation de la vitesse du flot d'air -- pour une micro-valve activée
● <i>signal Vox_s, Voz_s : real;</i>	
<i>signal fall : boolean;</i>	-- Choc de l'objet avec le sol
● <i>signal b1, b2, b3, b4, b5, b6 : boolean;</i>	-- Positions intermédiaires
● <i>signal n1,n2 : boolean;</i>	-- Positions intermédiaires
● <i>type states is (stage1, stage2);</i>	
● <i>signal state1 : states;</i>	-- Etat 1 : détection de l'extrémité droite de l'objet
● <i>signal state2 : states;</i>	-- Etat 2 : détection de l'extrémité gauche de l'objet

Fig. 4.20 – Déclaration de l'entité de la cellule « convoyage à 1-D en lévitation ».

Corps d'architecture

De la même manière que pour la déclaration d'entité, le corps d'architecture du modèle « convoyage à 1-D en lévitation » sera défini à partir de celui du modèle « lévitation : cas général » comme le montre la figure 4.21.

```

ARCHTECHTURE behavior Matrice is
.....
BEGIN
-- Initiation
● break Dx => Dx0, .....;
-- Calculs des forces
.....
● FC == 0.5*Dens*Cxp*Vaxoff**2*Sb;
-- Calculs des forces de résistance
.....
● -- Détection de l'extrémité de l'objet
.....
● -- Calculs de la surface où les forces fluidiques sont appliquées
.....
-- l'objet heurt le sol
.....
-- Relations de la dynamique
.....
● FC - FCr == Mass*Vox'dot;
● Vox == Dx'dot;
END;

```

Fig. 4.21 – Programmation VHDL-AMS du modèle « convoyage 1-D en lévitation ».

Machine d'états (FSM)

Nous devons établir aussi un sous-modèle de détection de l'objet. Ce sous-modèle doit nous permettre de détecter l'objet, de reconnaître sa forme et de localiser les cellules de la matrice qui sont couvertes par lui.

Pour remplir ces trois conditions, nous avons établi une procédure de détection des extrémités de l'objet au-dessus de la zone active du microactionneur pneumatique (micro-valve). Cette procédure a été appliquée à une cellule, puis distribuée sur l'ensemble des cellules de la matrice s'étalant de N à $(N + 4)$, comme le montre la figure 4.22.

Nous atteignons notre objectif de modélisation tout en réduisant la complexité de la future programmation, nous utilisons une simple machine d'états pour décrire cette fonction de détection. Ainsi, selon la position de l'objet en mouvement par rapport à la zone active de la micro-valve, nous affectons les valeurs ($= 0$) et (> 0) aux deux états ($S1$) et ($S2$), tels que :

- $S1$: L'objet se déplace vers la gauche. Quand l'extrémité de l'objet pénètre dans la zone

d'exposition de la micro-valve, l'état S_1 se modifie en passant de la valeur $S_1 = 0$ à $S_1 > 0$, et quand il sort de cette même zone, l'état revient à sa valeur d'origine $S_1 = 0$.

- S_2 : L'objet se déplace vers la droite. Quand l'extrémité de l'objet pénètre dans la zone d'exposition de la micro-valve, l'état S_2 se modifie en passant de la valeur $S_2 = 0$ à $S_2 > 0$, et quand il sort de cette même zone, l'état revient à sa valeur d'origine $S_2 = 0$.

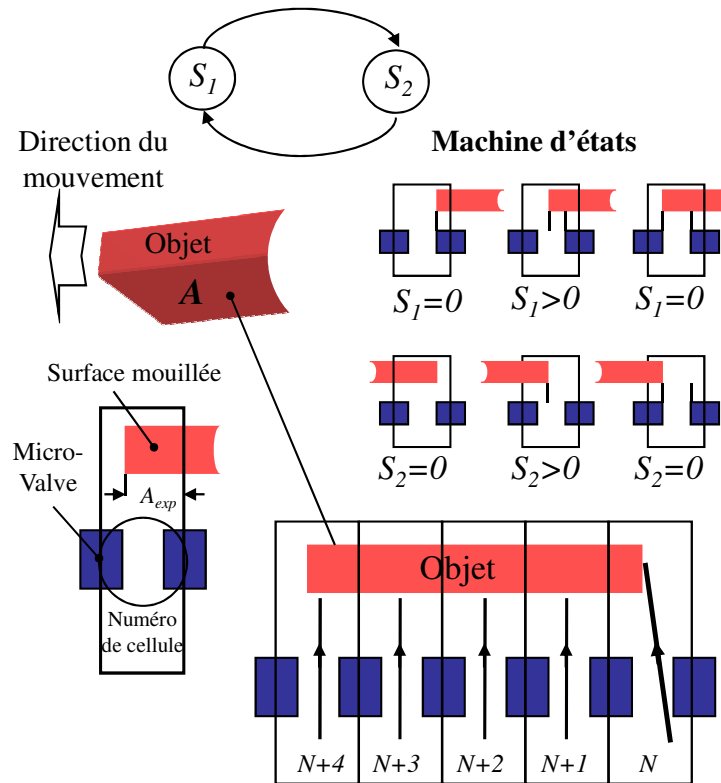


Fig. 4.22 – Modèle de détection distribuée de l'objet sur la matrice.

4.4.5 Programme du modèle « Convoyage 2-D en lévitation »

4.4.5.1 Éléments de modélisation

Le modèle « convoyage à 2-D en lévitation » se différencie très peu du modèle précédent « convoyage à 1-D en lévitation », dont nous reprendrons la cellule de base dans ce modèle. Cependant, l'arrangement des cellules du dispositif réel de micromanipulation va nous obliger à quelques modifications de description.

La figure 4.22 donne la représentation du réseau de microactionneurs pneumatiques du dispositif réel de micromanipulation. On notera que la cellule de base du modèle « convoyage à 1-D en lévitation » est reprise dans cette configuration en alternance avec des micro-valves horizontales et verticales, comme le montre la figure 4.23. Nous pouvons considérer que le modèle « convoyage à 1-D en lévitation » décrit le comportement du système sur l'axe. Maintenant, nous ajoutons l'axe (y) dans le modèle.

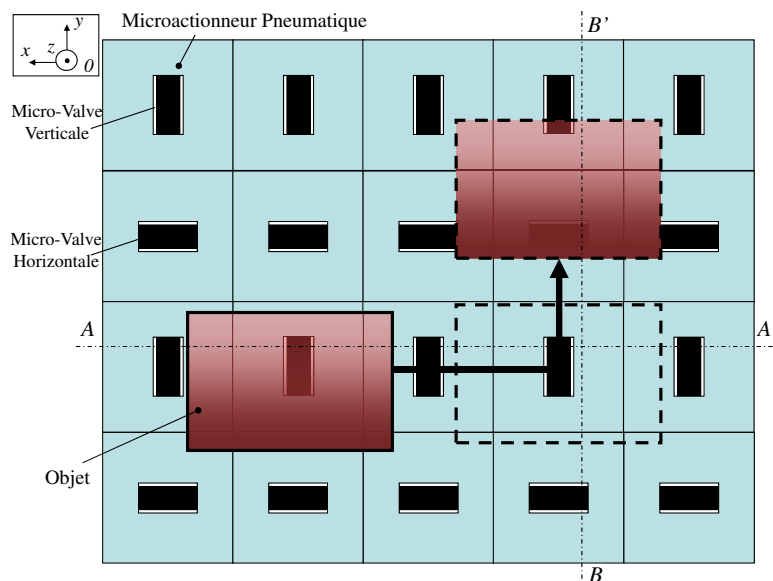


Fig. 4.23 – Description du réseau de microactionneurs pneumatiques du dispositif réel de micromanipulation (vue de dessus).

Cette ergonomie ne change pas l'approche de description du convoyage à 1-D que nous avons élaborée précédemment. Par contre, la description du convoyage à 2-D en sera modifiée, comme l'illustre les deux représentations de la figure 4.24. En effet, si le convoyage sur l'axe (x) entre les points A et A' ne change pas de la description du convoyage à 1-D, il n'en sera pas de même lorsque l'objet sera convoyé sur l'axe (y) et les points B et B'. Dans ce dernier cas, l'espace entre deux cellules se modifie. Il faudra donc en tenir compte dans notre description finale.

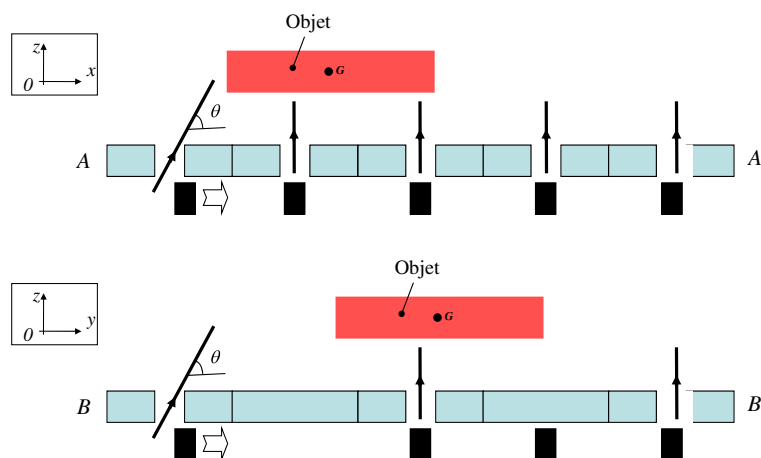


Fig. 4.24 – Description du réseau de microactionneurs pneumatiques du dispositif réel de micromanipulation (vue en coupe).

4.4.5.2 Éléments de description

Le principe de la construction de notre programme se base sur une modification du précédent code VHDL-AMS de la cellule « convoyage 1-D en lévitation ». En effet, nous avons observé que cette dernière peut être réutilisée pour construire la cellule « convoyage 2-D en lévitation ». La figure 4.25 montre la démarche de construction adoptée. Les principes sont établis en trois étapes :

1. On forme une matrice de microactionneurs verticaux (1) avec plusieurs cellules « convoyage 1-D en lévitation » modifiées pour créer l'espace d'arrangement désiré ;
2. On fait pivoter de 90° cette matrice pour obtenir une matrice de microactionneurs horizontaux (2) ;
3. On insère les cellules « convoyage 1-D en lévitation » d'origine dans la matrice obtenue à l'étape 2.

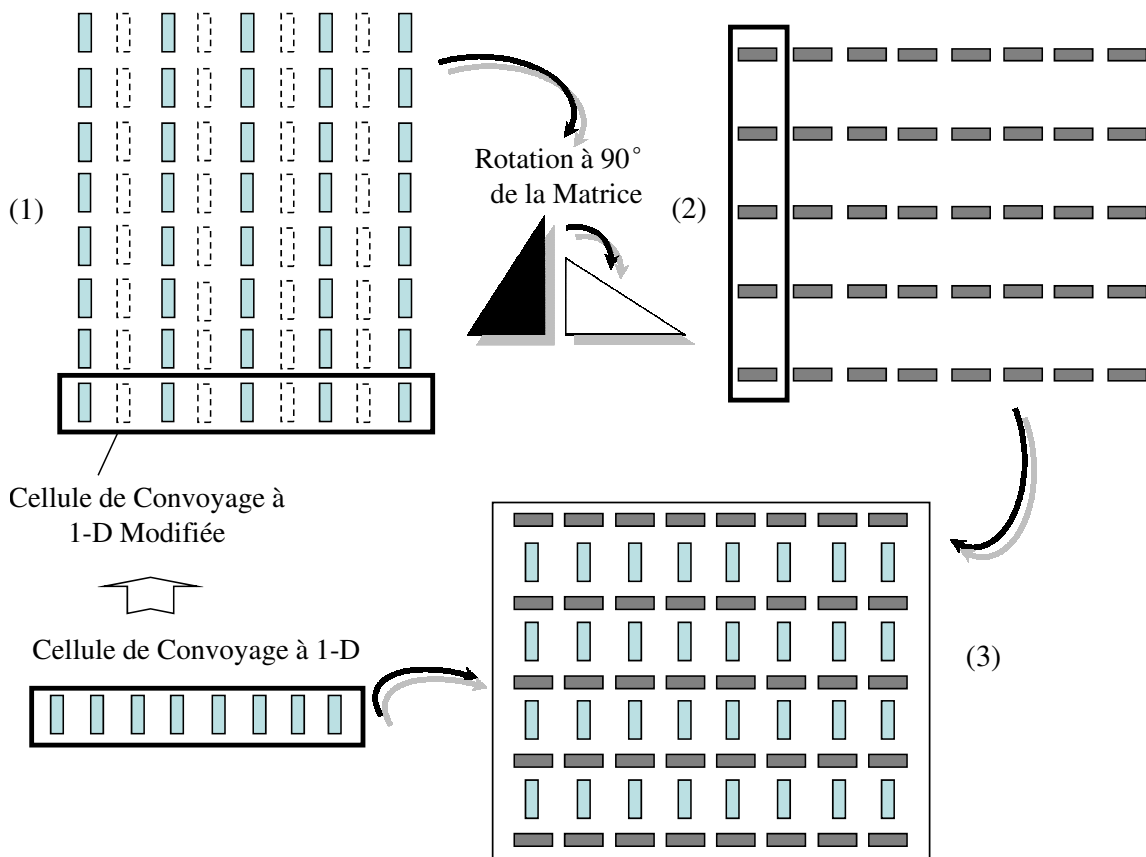


Fig. 4.25 – La construction du modèle « convoyage 2-D en lévitation »

Avec ce dispositif, nous allons adopter quelques principes de fonctionnement du modèle :

- La matrice (1) de la figure 4.25 ne s'occupe que du déplacement de l'objet sur l'axe (x) ;
- La matrice (2) de la figure 4.25 ne s'occupe que du déplacement de l'objet sur l'axe (y) ;
- Il n'y a pas d'interférence entre les deux matrices.

Les principes de fonctionnement que nous avons adoptés permettent une simplification de la description VHDL-AMS du modèle. Les phénomènes pneumatiques sur l'axe (y) sont tout à fait les mêmes par rapport à l'axe (x). Nous pouvons réutiliser les mêmes descriptions sur l'axe (x) sans beaucoup de modifications sur la description de l'axe (y).

L'ensemble du code VHDL-AMS du programme du modèle « convoyage 2-D en lévitation » est donné en Annexe.

Abordons à présent les aspects « simulation » des modèles de la matrice dans les cas de convoyage à 1-D et 2-D. Une confrontation avec les données expérimentales, obtenues par nos partenaires de recherches, sera également envisagée.

4.5 Simulation

4.5.1 Outils et paramètres

Comme il en a été discuté dans le Chapitre 3, l'ensemble des simulations que nous présentons dans cette section a été réalisé avec le logiciel SMASHTM de la société Dolphin Integration®.

Le choix des conditions de simulation a été établi à partir des données expérimentales que nous connaissons grâce à nos partenaires de recherche de l'Université de Tokyo, qui ont validé en premier la fonctionnalité du dispositif de micromanipulation à base de microactionneurs pneumatiques. Le tableau 4.4 donne l'ensemble des paramètres de simulation que nous appliquerons.

Paramètres	Description	Valeur	Unité
m_o	Mass de l'objet	$6,6 \times 10^{-6}$	kg
t_o	Épaisseur de l'objet	$2,5 \times 10^{-6}$	m
w_o	Largeur de l'objet	$4,5 \times 10^{-3}$	m
L_o	Longueur de l'objet	$4,5 \times 10^{-3}$	m
C_{xp}	Coefficient de pression	1,11	
C_{xf}	Coefficient de frottement	0,004	
ρ	Densité de l'air	1,3	kg/m^3
v_a	Vélocité du flot d'air	5,0	m/s

Tab. 4.4 – Paramètres de simulation

4.5.2 Cas du « convoyage 1-D en lévitation »

La figure 4.26 donne les résultats de simulation du modèle « convoyage 1-D en lévitation ». Rappelons que la simulation décrit le parcours d'un objet sur les cinq premières cellules de la surface active. Les courbes de résultat décrivent respectivement (dans l'ordre de haut en bas) :

1. La hauteur de l'objet (D_z) : la hauteur de l'objet est proportionnelle à la vitesse du flux d'air en repos. Étant donné que cette vitesse reste constante à tout moment, l'objet garde

la même hauteur tout au long de la simulation.

2. La vitesse du jet d'air (V_{ax}) : c'est la vitesse de la composante en (x) du jet d'air générée par le microactionneur pneumatique lorsque la micro-valve est activée. Cette composante représente la force de poussée qui s'applique sur la tranche arrière de l'extrémité de l'objet. Nous constatons que ce jet d'air ne s'établit, comme prévu, que de manière ponctuelle au moment de la détection de l'extrémité de l'objet.
3. La force de convoyage (F_C) : c'est la force de convoyage appliquée sur l'objet et qui est dépendante de la vitesse du jet d'air décrite sur la courbe précédente, ce qui est vérifié par la simulation. En effet, la force n'apparaît que lorsque la vitesse du jet d'air s'établit.
4. La vitesse de convoyage de l'objet (V_{ox}) : à chaque nouvelle poussée, générée par la force de convoyage du microactionneur pneumatique, la vitesse de l'objet est modifiée et augmente brusquement avant de s'établir et même décroître avant une nouvelle séquence. Les simulations montrent de manière très nette ce phénomène de progression de la vitesse de l'objet, chose qu'il est très difficile à mesurer expérimentalement.
5. L'accélération du convoyage de l'objet.
6. La résistance de l'air (F_{Cr}) : la résistance de l'air est proportionnelle au carré de la vitesse de l'objet. C'est pourquoi la courbe de la résistance d'air est similaire à celle de la vitesse de l'objet sur nos résultats de simulation.

Les résultats de simulation que nous avons obtenu valident la fonctionnalité du « modèle comportemental » à partir du langage VHDL-AMS de la matrice. Ces premiers résultats de validité seront repris et confirmés dans différentes conditions de simulation.

Remarque :

Il est intéressant de noter également que si nous constatons que la dynamique de l'objet augmente lors des actions de poussée de chaque microactionneur pneumatique, la vitesse des jets d'air, elle reste inchangée. Cette observation est très importante au niveau de notre description car cela signifie que toutes les cellules ont le même comportement, et il est indépendant de la position de l'objet.

Cela prouve que les cellules agissent aussi comme si elles étaient indépendantes alors que dans le modèle comportemental, le comportement de la surface active est décrit de manière globale, c'est-à-dire que les cellules ne peuvent pas être traitées individuellement. C'est un atout supplémentaire de notre modèle.

4.5.3 Convoyage 2-D en lévitation : Cas en boucle ouverte

La figure 4.27 montre les résultats de simulations du modèle « convoyage 2-D en lévitation » dans un cas en boucle ouverte. Pour juger des performances de notre modèle, nous avons essayé de reproduire le parcours d'un objet lors d'une phase expérimentale. Cette dernière est décrite ici en quatre séquences de convoyage correspondant à une prise vidéo du convoyage de l'objet

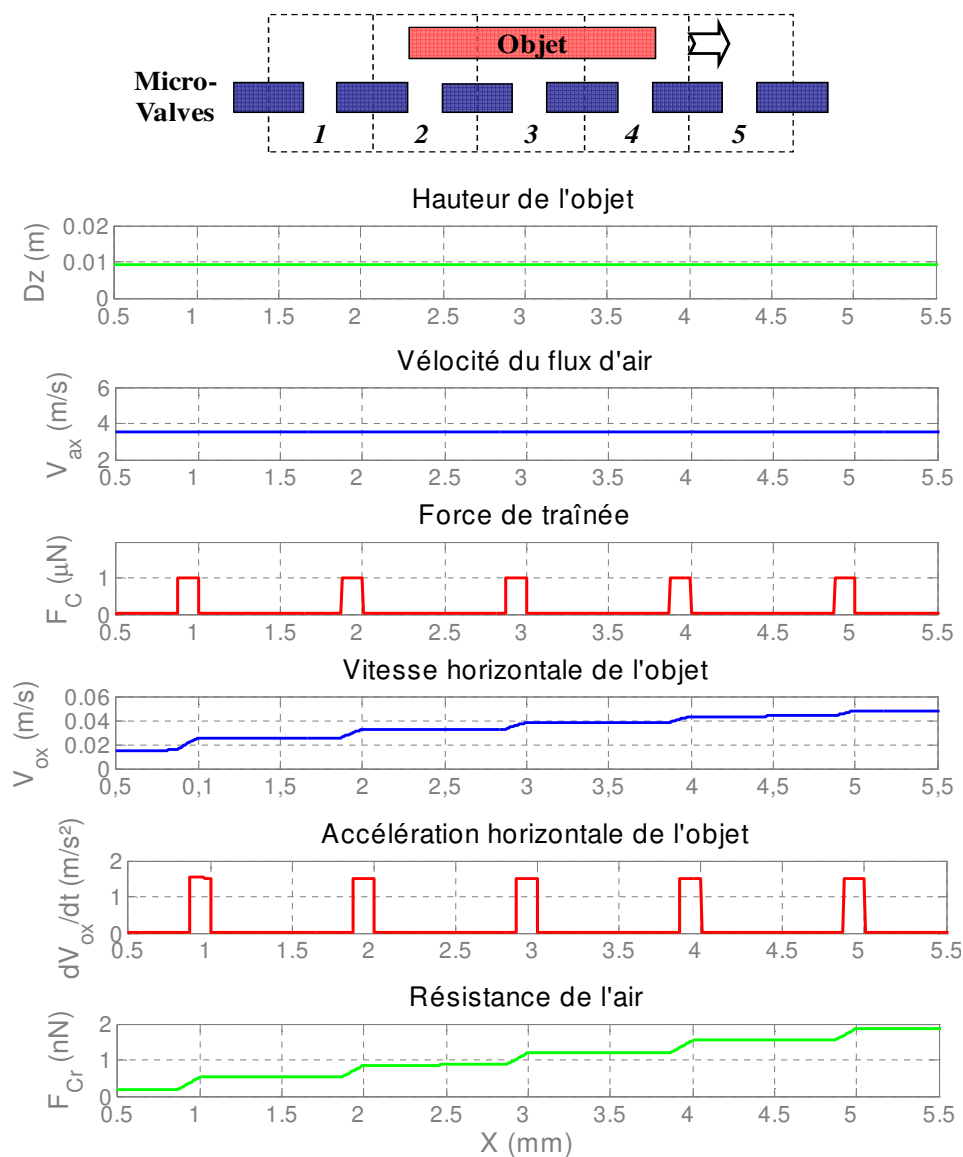


Fig. 4.26 – Résultats de simulation du « convoyage sur 1-D en lévitation ».

en boucle ouverte vers le centre de la matrice.

Le point de départ de l'objet se trouve en bas à gauche de la matrice de microactionneurs pneumatiques que l'on voit sur la séquence (1) de la figure 4.27(a). Au démarrage de la simulation, la vitesse initiale de l'objet est donnée dans le sens positif de l'axe (y), ce qui est supposé représentatif de la situation expérimentale (difficulté de connaître l'état de l'objet au moment de son démarrage sur la vidéo). Ensuite l'objet est convoyé légèrement en diagonale soit sur les deux axes (x) et (y) ce qui est bien repris sur la simulation.

Lors du changement de direction de l'objet (convoyage 2-D), on observe bien, dans les deux cas de figure (expérimentation et simulation), que la poussée d'origine vers l'est de la matrice est soudainement modifiée avec une poussée vers le nord entraînant la déviation de l'objet.

Au niveau des dimensions de convoyage la similitude entre les résultats expérimentaux et de simulation est en tout point remarquable, même si comme nous pouvions nous y attendre, il est pratiquement impossible de reproduire à l'identique la phase expérimentale de convoyage sachant que l'application de jet d'air de poussée sur l'objet en réalité est bien souvent très aléatoire. En effet, une faible différence de distance entre l'extrémité arrière de l'objet et le jet d'air appliqué, lorsque la micro-valve est activée, peut avoir des effets fluidiques allant du simple à dix fois l'amplitude du flux d'air appliqué. C'est en grande partie pour ces raisons que ce type de dispositif requière un système de contrôle en boucle fermée, ce qui fera l'objet du prochain chapitre de ce mémoire.

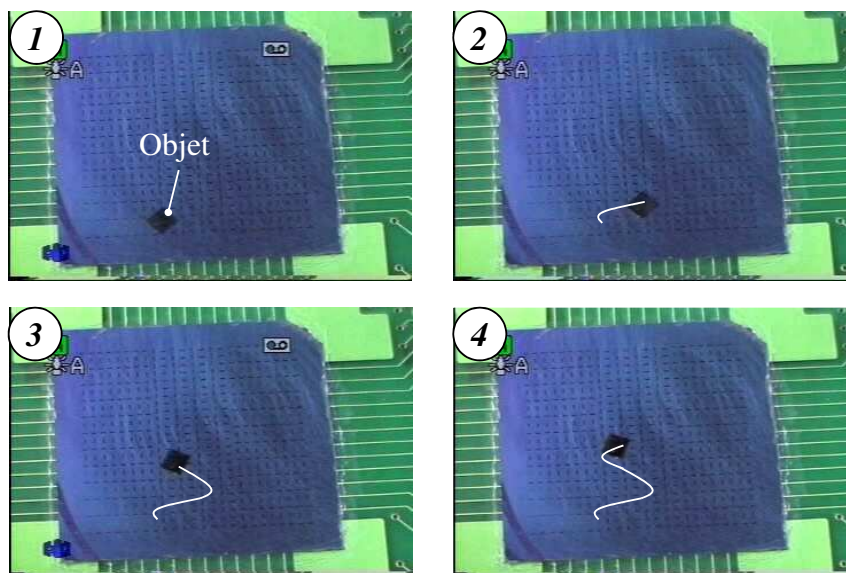
4.6 Conclusion et perspective

Dans ce Chapitre 4, nous avons établi avec succès le « modèle comportemental » de la matrice de microactionneurs pneumatiques, qui correspondra au modèle décrit au plus haut niveau d'abstraction de notre hiérarchie de modélisation.

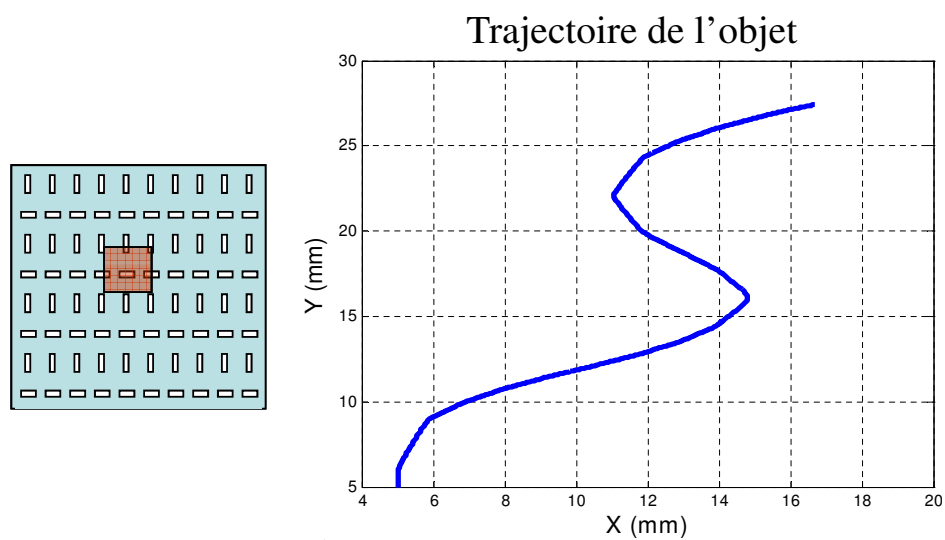
Après avoir déterminé tous les éléments du modèle de l'environnement fluide de notre dispositif, nous nous sommes appliqués à modéliser ce dernier en suivant une arborescence très précise. Cette approche de description nous permettra de construire notre modèle en utilisant une structure à « brique » réutilisable, ce qui s'adaptera très facilement avec l'établissement du code VHDL-AMS lors de l'élaboration du programme du modèle. Nous noterons à cet effet, la grande flexibilité de ce langage.

Les résultats de simulation du « modèle comportemental » de la matrice viendront corroborer les attentes fonctionnelles lors d'un convoyage à 1-D d'un objet. De manière plus spectaculaire, nous réussirons à reproduire par simulation la trajectoire, obtenue expérimentalement en boucle ouverte, d'un objet convoyé en lévitation sur la surface active de la matrice.

Cette première phase de modélisation ayant permis l'établissement et la validation du modèle « global » de notre dispositif, il nous servira de référence lors de nos prochains développements comme le préconise l'approche de « Prototypage Virtuel Fonctionnel » que nous adoptons ici.



(a)



(b)

Fig. 4.27 – Résultats de simulation du « convoyage sur 2-D en lévitation : Cas en boucle ouverte » et comparaison avec les données expérimentales

Dans les travaux futurs, nous travaillerons aussi sur la modélisation 3-D avec des rotations de l'objet dans les trois directions, et le cas d'objets de forme quelconque, de taille quelconque.

Chapitre 5

« Modèle structurel comportemental » et « modèles composants »

5.1 Approche de modélisation

5.1.1 Préambule

Le « modèle comportemental » développé lors de l'étude précédente nous a donné une description globale du système. Ce modèle représente une vue « portée de l'extérieur », grâce à laquelle nous pouvons connaître les liaisons entre le système et l'environnement où il se produit. Il permet aussi de connaître les influences de l'environnement au système et vice versa. Par contre, à ce niveau de description, nous n'avons pas une vue « portée de l'intérieur ». C'est-à-dire avec le système vu comme un assemblage de sous-systèmes ou composants interconnectés [59]. C'est ce que décrivent précisément le « modèle structurel comportemental » et à un niveau d'abstraction plus bas, les « modèles composants ».

Dans la construction de ces derniers modèles, on s'attache à décrire la structure d'un système sous forme d'arbre en décrivant les interconnexions entre différents éléments ou sous-ensembles définis individuellement. Chaque élément ou sous-ensemble peut alors être caractérisé par une description « comportementale » ou raffiné par une autre représentation « structurelle ».

Pour élaborer de tels modèles en respectant l'approche de modélisation que nous nous sommes fixée dans cette thèse, il nous faudra répondre à plusieurs questions. Comment les construire tout en suivant l'approche de conception basée sur la méthode de « Prototypage Virtuel Fonctionnel » précédemment évoquée ? Comment passer du « modèle comportemental » au « modèle structurel comportemental » en adoptant cette démarche ? Comment extraire les éléments du « modèle structurel comportemental » et atteindre le niveau des « modèles composants » ?

Pour répondre à toutes ces questions nous établirons un flot de modélisation que nous présenterons dans la section suivante.

5.1.2 Flot de modélisation

5.1.2.1 Rôle des spécifications

L'approche de modélisation du « modèle structurel comportemental » doit être étroitement liée aux spécifications « fonctionnelles » et « technologiques » du dispositif à concevoir. Nous pouvons envisager deux méthodes de modélisation liées aux spécifications de conception.

Si ces spécifications existent déjà et que l'objectif de la modélisation est de les améliorer sur le plan des performances et de l'économie, alors on utilisera une méthode de modélisation dite « méthode schéma » signifiant que chaque composant du modèle devra être décrit comme une structure physique. Dans ce cas, il est alors nécessaire de modéliser le système en respectant tous les détails existants, incluant par exemple les normes de conception. Grâce à un tel niveau de description, de nombreux phénomènes, impossible à observer expérimentalement, sont visualisables par simulation. Ainsi, les performances du système peuvent être améliorées ou ses contraintes corrigées. Cette approche de modélisation s'est inspirée des travaux de *Z. Toffano et al.* lors de l'élaboration du projet Shaman [87].

Dans l'autre cas dit de la « méthode du cahier des charges », on ne connaît pas la physique du dispositif final mais seulement le « cahier des charges » de conception. Dans cette approche plutôt classique, l'objectif de modélisation est de choisir une structure, une technologie ou encore un matériau permettant d'atteindre les meilleures performances dictées par le « cahier des charges ». Ainsi, l'approche de modélisation « structurelle » basée sur le « modèle comportemental » devient beaucoup plus libre, facilitant ainsi des études plus profondes du système. Ceci permet ensuite d'aider le concepteur à optimiser son système en étudiant plusieurs choix de composants et de connexions.

C'est sur cette « méthode du cahier des charges » que nous fonderons notre approche de modélisation. Abordons à présent les étapes qui nous conduiront à la modélisation du « modèle structurel comportemental ».

5.1.2.2 Étapes de modélisation

La clé de la modélisation du « modèle structurel comportemental » est de permettre, dès l'élaboration du « modèle comportemental », l'identification de sous-modèles constituant ce dernier et semblables au sens « comportemental » du terme. Tous ces sous-modèles semblables à des « modèles comportementaux » devront alors se retrouver dans le modèle « structurel » sous forme de « composant ». Aucun élément nouveau du modèle d'origine ne devant être introduit.

La procédure de la construction est ensuite basée sur l'établissement des interconnexions existantes entre les différents composants. Ces interconnexions devront permettre de transférer

5.1.3 Interface

5.1.3.1 Éléments multi-domaines et interface de contact

Du point de vue physique, en étudiant les composants de types MEMS, on doit prendre en compte à la fois des grandeurs électriques (tension, courant, etc.) mais également d'autres grandeurs physiques comme la température, la vitesse, etc. De tels composants sont ainsi appelés « éléments multi-domaines » car ils échangent des informations de différents domaines physiques.

Pour décrire ces échanges multi-domaines, nous définirons la notion « d'interface de contact » qui n'est autre qu'un concept pour traiter l'interaction existante entre deux types de grandeurs physiques [58]. « L'interface de contact » sert comme d'interprète entre deux grandeurs physiques.

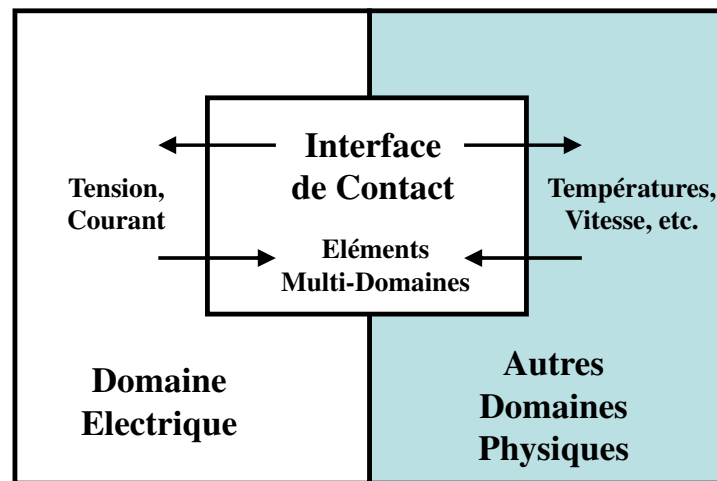


Fig. 5.2 – Représentation des « éléments multi-domaines » et du concept « d'interface de contact ».

La figure 5.2 montre les interactions existantes entre deux domaines physiques lesquels se produisent par l'intermédiaire d'un élément « multi-domaine » (une partie du système qui appartient simultanément à deux ou plusieurs domaines différents).

Cependant, il serait intéressant de voir comment traiter ce concept « d'interface de contact » dans le cas de la construction du « modèle structurel comportemental ». Nous proposerons dans ce cas de créer un composant dit composant « interface ».

5.1.3.2 Le composant « interface »

Le composant « interface » est un composant virtuel que nous allons créer dans le but de relier entre eux deux composants dont les domaines physiques sont incompatibles et ne peuvent être traduits que par une « interface de contact ». En créant le composant « interface », on évitera toute transformation des deux composants à interconnecter.

L'avantage de cette solution est qu'elle permet de conserver la physique de chaque composant qui est donc maintenu parfaitement intact. Si le modèle s'en trouve alourdi, il y gagne indéniablement en lisibilité. C'est ce que nous étudierons dans le cas de la « surface active » où le composant « interface » servira à décrire la physique des forces appliquées à l'objet.

Finalement, l'insertion du composant « interface » dans le « modèle structurel comportemental » peut se faire de deux manières :

- Soit on intègre le composant « interface » dans l'un des composants concernés dans les échanges d'information,
- Soit on crée un nouveau composant qui est indépendant des autres dispositifs physiques du système.

En dehors de l'intérêt de cette solution qui s'adapte très bien à notre méthode de modélisation « structurelle », nous allons aussi montrer que le langage VHDL-AMS s'adapte parfaitement à cette approche de modélisation du composant « interface ».

5.2 Modèle structurel comportemental

Appliquons à présent nos études de construction au « modèle structurel comportemental » de la matrice de microactionneurs pneumatiques.

Nous proposons également d'étudier la dimension multifonction de la matrice. Ainsi, suite au premier modèle de la matrice, nous intégrons dans le modèle les capacités d'autonomie (capteur) et d'intelligence (contrôleur) de cette dernière. Plus spécifiquement, nous parlons de « smart surface » pour évoquer cette matrice. Nous détaillerons dans les sections suivantes les trois étapes de construction constituant le flot de modélisation de la « smart surface », que nous présentons dans la figure 5.3.

5.2.1 Étape « identification »

En analysant le « modèle comportemental » du système, nous voyons que ce dernier décrit les comportements statiques et dynamiques d'un objet sur une surface pneumatique qui se compose d'un certain nombre de cellules identiques. Le système utilise des signaux mixtes et présente des caractéristiques multi-domaines.

Dans ce modèle, les comportements et les échanges de toutes les parties sont traités dans un même « composant ». Nous constatons cependant que la fonctionnalité du modèle peut être identifiée par les deux sous-modèles suivants :

- Le sous-modèle « surface active »,
- Le sous-modèle « objet ».

Les interactions entre ces deux sous-modèles se produisent à travers les jets d'air générés par la « surface active », comme le montre la vue « système » de la figure 5.4. A l'image du « modèle comportemental », nous pouvons considérer les sous-modèles comme des composants.

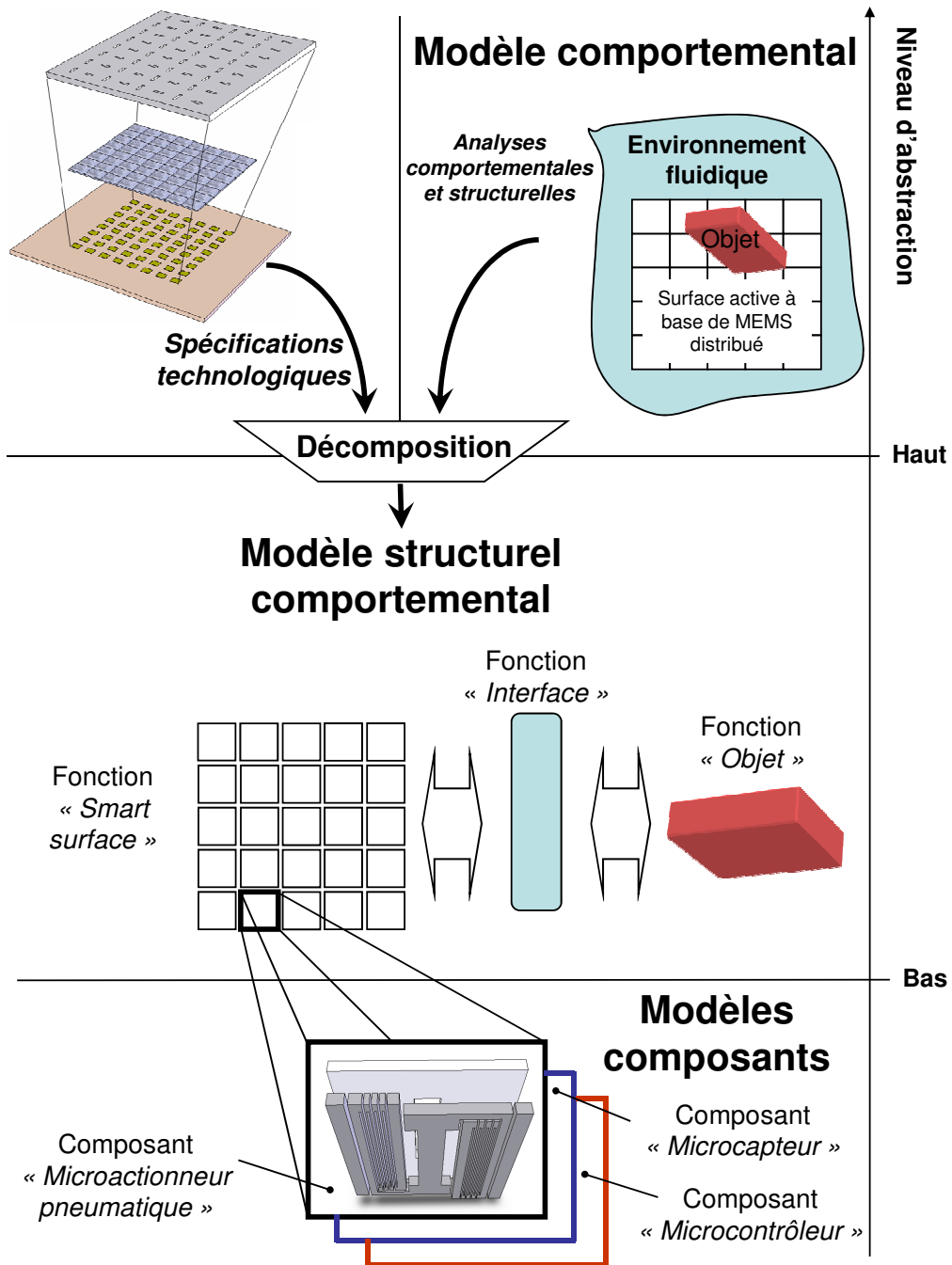


Fig. 5.3 – Flot de modélisation de la matrice transformable en « smart surface ».

Le composant « surface active » comme une simple « boîte noire » avec uniquement un signal de déclenchement en entrée et une grandeur représentant la « vitesse » des jets d'air sortants. Toutes ces informations peuvent donc être traitées à partir de signaux discrets, comme le montre la vue « modèle » de la figure 5.4. Quant au composant « objet », les informations qui le caractérisent telles que la vitesse, l'accélération, la hauteur, la position (..) sont bien indépendantes du composant « surface active ». Ce sont des signaux exclusivement continus qui évoluent dans le temps.

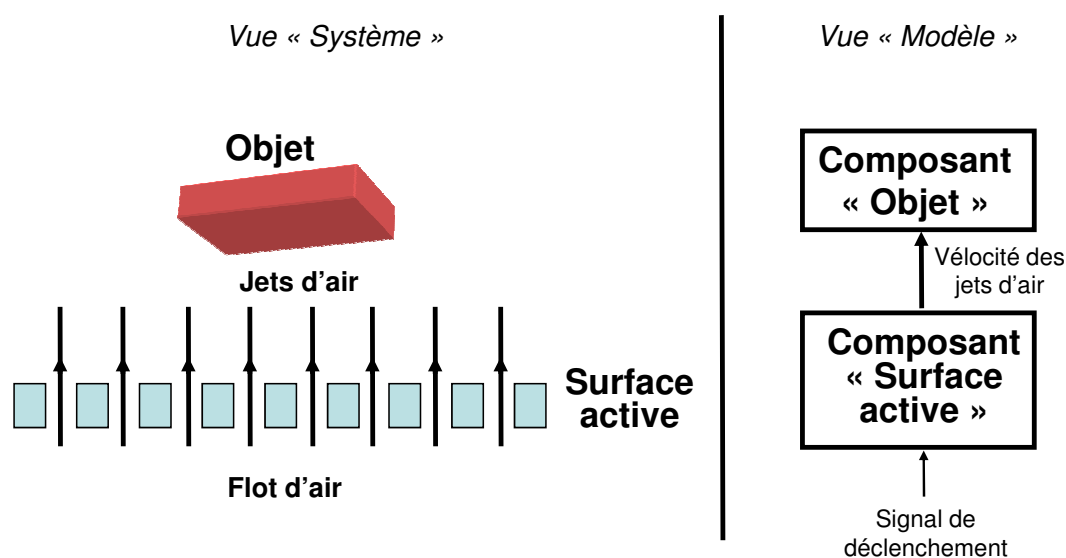


Fig. 5.4 – « Modèle structurel comportemental » à l'étape d'identification.

5.2.2 Étape « décomposition »

5.2.2.1 Décomposition de la « surface active »

En fait, une décomposition du « modèle comportemental » à deux composants, comme proposée dans l'étape d'identification précédente, ne pourrait nous satisfaire car le « modèle structurel comportemental » en résultant ne répondrait pas aux critères multi-physiques inhérents à cette description. En effet comment interconnecter les composants « objet » et « surface active » qui sont des composants de deux domaines physiques différents.

Il nous faudra créer un composant « interface » qui, comme introduit précédemment, permettra de maintenir séparé et isolé les composants « surface active » et « objet ». Le rôle du composant « interface » sera alors d'interpréter les informations reçues par le composant « surface active », générant les jets d'air, avant de les transmettre au composant « objet », et vice versa, comme le montre la figure 5.5.

A noter que les informations transmises au composant « objet » proviennent du composant « microactionneur », lequel génère les forces fluidiques appliquées au composant « objet ». On voit de cette manière que la « surface active » va être décomposée en deux composants : « l'inter-

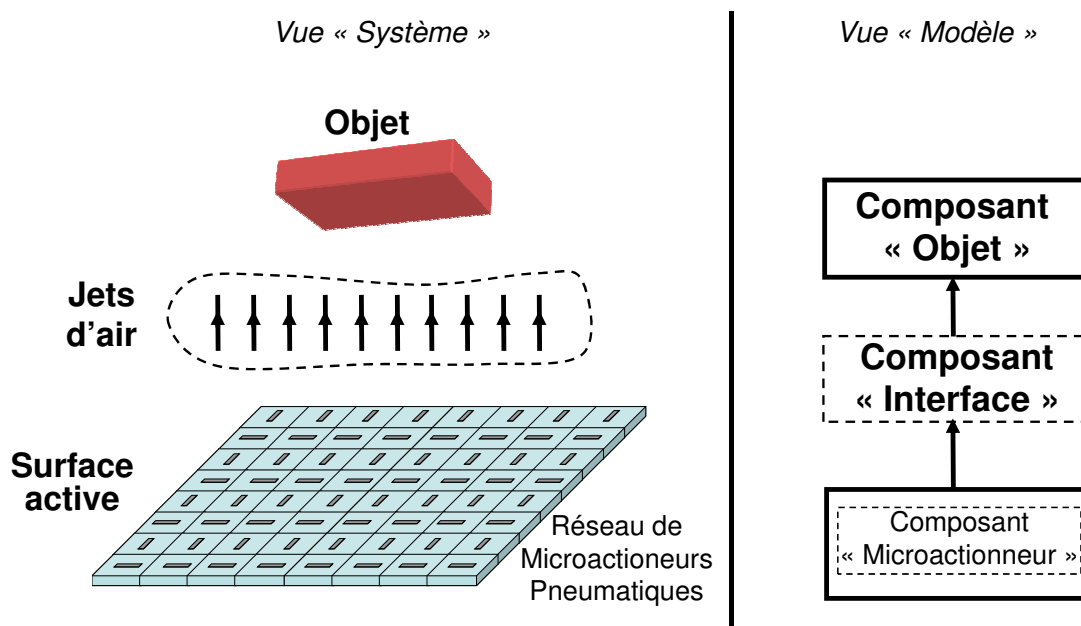


Fig. 5.5 – « Modèle structurel comportemental » de la « surface active ».

face » et le « microactionneur ». Ce dernier sera ensuite démultiplié pour constituer une matrice de composants « microactionneur » que nous appellerons : « couche microactionneur ». Cette matrice représentera dans la réalité un réseau de microactionneurs, comme décrit sur la figure 5.5.

On rajoutera que si nous désirons créer un nouveau composant « surface active » utilisant les principes d'un système « à contact » pour déplacer l'objet au lieu du flux d'air, ce qu'il faut modifier dans notre modèle structurel est le composant « interface », nous remplaçons les équations de l'aérodynamique par les équations décrivant les phénomènes de contact physique pour calculer les forces appliquées au composant « objet ». Alors que nous n'avons pas besoin de modifier le composant « objet », car les informations qu'il reçoit du composant « interface » sont uniquement des grandeurs de forces, il ne nécessite pas de connaître l'origine de ces forces pour calculer ses propres mouvements.

La figure 5.6 présente l'exemple d'un système « à contact ». Les jets d'air de la matrice à base de microactionneurs pneumatiques sont remplacés par des « pieds-mobiles » en polymère actionnés par échauffement pour créer une déformation locale.

5.2.2.2 Décomposition de la « smart surface »

Si nous désirons aller encore plus loin dans le « modèle structurel comportemental », nous pouvons proposer de décomposer le composant « surface active » à partir du composant « interface » et des trois composants de base constituant la « smart surface », soit :

- Le composant « microactionneur »,

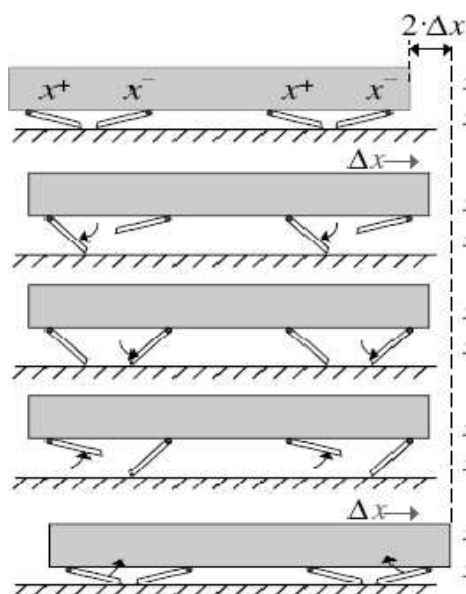


Fig. 5.6 – Système de micromanipulateur « à contact ».

- Le composant « microcapteur »,
- Le composant « microcontrôleur ».

Ces trois composants n'en forment qu'un, qui ensuite sera démultiplié pour permettre la constitution d'une matrice de ce composant qui constituera la « smart surface » à partir de trois couches : la couche « microactionneurs », la couche « microcapteurs », la couche « microcontrôleurs ».

A noter que le composant « smart surface » pourra être interprété dans la réalité par trois réseaux parfaitement juxtaposés : le réseau microactionneurs, le réseau microcapteurs, le réseau microcontrôleurs.

La figure 5.7 présente la structure du « modèle structurel comportemental » après l'étape de décomposition basé sur le « modèle comportemental ».

5.2.3 Étape « interconnexion »

Après avoir choisi la décomposition du « modèle structurel comportemental » et l'ensemble des composants qui le composent, nous devons maintenant établir le réseau de communication entre chaque composant. Plus précisément, nous devons établir les « entrées-sorties » de chaque composant du système. Pour ce faire, il faudra d'abord définir le rôle exact de chaque composant, puis analyser ses liens avec les autres composants pour lui donner finalement les « entrées-sorties » appropriées.

La figure 5.8 reprend la structure générale du « modèle structurel comportemental » que nous allons adopter dans ce travail d'interconnexion.

Présentons et expliquons à présent les interconnexions existantes entre chacun des compo-

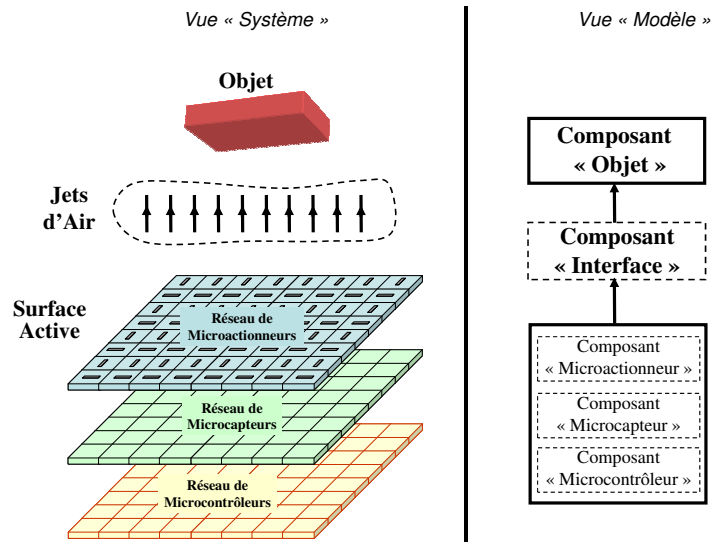


Fig. 5.7 – « Modèle structurel comportemental » de la « smart surface ».

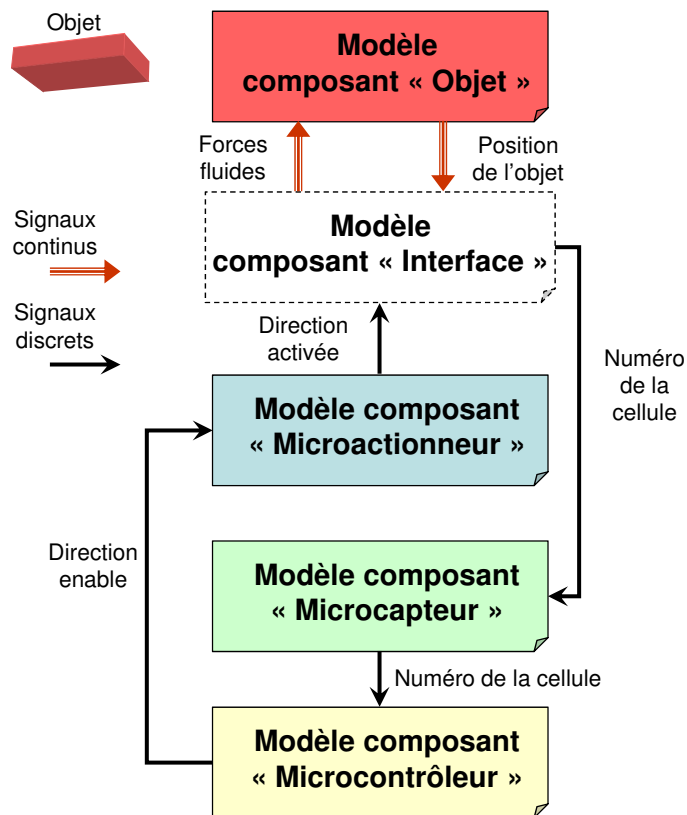


Fig. 5.8 – « Modèle structurel comportemental » suite à l'étape d'interconnexion.

sants.

5.2.3.1 Composant « objet »

Il reçoit des informations du composant « interface ». En fonction des forces qui s'appliquent et les caractéristiques physiques du composant « objet », il calcule l'accélération, la vitesse, la hauteur, la position de ce dernier. Sa position sera transférée au composant « microcapteur ».

5.2.3.2 Composant « interface »

Il relie les composants « surface active » et « objet ». Son rôle est d'interpréter les informations reçues par le composant « surface active » et de les transmettre au composant « objet », et vice versa.

Les informations transmises au composant « objet » proviennent du composant « microactionneur » lequel génère les forces fluidiques appliquées au composant « objet ». Parmi les informations fournies par le composant « microactionneur », on trouve la « vitesse » du flux d'air à l'état « repos » et « actif » de la micro-valve inhérente à sa structure. En fonction de ces informations, le composant « interface » calculera les différentes forces fluidiques apportées via les jets d'air. Ces forces sont transmises au composant « objet » sous forme de signaux continus.

Inversement, le composant « objet » transmet l'information de la position de l'objet au composant global « surface active » via le composant « microcapteur ». L'information se présentera sous forme discrète puisqu'il s'agira uniquement des coordonnées (ou numéros) de la matrice de microactionneur que l'objet est en train de couvrir.

5.2.3.3 Composant « surface active »

Composant « microactionneur »

Il reçoit une consigne du composant « microcontrôleur » qui lui indique s'il est actif ou au repos. Actif, il doit communiquer au composant « objet », l'information concernant le jet d'air qui produit, soit l'amplitude et la direction de sa vitesse. L'information de consigne est donnée en signaux discrets et doit être interprétée et traduite en signaux continus par l'intermédiaire du composant « interface » au composant « objet ».

Composant « microcapteur »

Il reçoit des informations indiquant la position du composant « objet ». Ces informations sont données en signaux continus, elles doivent être interprétées et transférées en signaux discrets par l'intermédiaire du composant « interface ». Ces informations seront ensuite transmises au composant « microcontrôleur ».

Composant « microcontrôleur »

Il reçoit des informations du composant « microcapteur » concernant la position du compo-

sant « objet », puis prend la décision en fonction de ces informations pour la transmettre au composant « microactionneur » en signaux discret via le composant « microcontrôleur ».

5.2.4 Quadrillage

Dans le « modèle comportemental », la « surface active » se comportait comme un seul composant, nous ne pouvions distinguer individuellement le comportement des cellules qui la composaient. À l'inverse, dans le « modèle structurel comportemental », le comportement de chacune des cellules de la « surface active » doit et peut désormais être décrit de manière individuelle. Étant donné que les cellules sont toutes identiques au niveau du composant « surface active », nous procéderons à un quadrillage identique sur la couche « microactionneur » qui la compose, comme le montre la figure 5.9(1) pour un nombre indicatif de cellules.

Dans le cas de la description de la « smart surface », nous obtenons un nombre de cellules possédant chacune les trois composants : « microactionneur », « microcapteur » et « microcontrôleur ». Chacune des cellules a un numéro correspondant à sa position dans la matrice de la « surface active ». Elles se comportent toutes de la même manière en fonction de signaux entrées. Chaque cellule étant numérotée, elles connaissent donc leur propre position, c'est une information essentielle dans une stratégie de contrôle.

On verra que la réalisation du quadrillage de la « surface active » est très simple à programmer à partir du langage VHDL-AMS. En effet, ce dernier possède l'instruction « Generate » qui nous permet de générer automatiquement une matrice du type de celle que nous souhaitons décrire ici. Nous reviendrons sur cette particularité de programmation dans la section suivante.

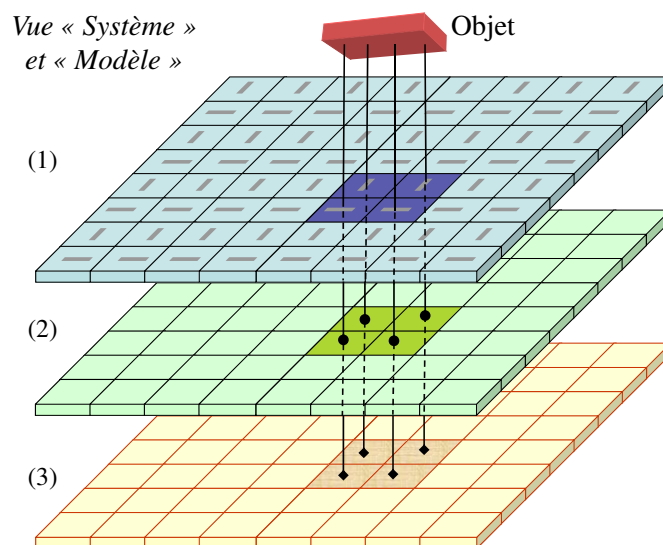


Fig. 5.9 – Quadrillage des trois couches de la « smart surface »

5.2.5 Description VHDL-AMS

5.2.5.1 Approche par unités de conception

Parmi les cinq unités de conception du « modèle structurel comportemental » de la « smart surface », seuls le corps d'architecture et la déclaration de configuration seront à modifier. Les unités de conception : déclaration d'entité, déclaration de paquetage et corps de paquetage, resteront identiques à celles que nous avons établies précédemment.

La figure 5.10 illustre la structure générale du programme VHDL-AMS du « modèle structurel comportemental » élaboré. On note que les cinq « composants », définis précédemment lors de l'approche de description du modèle, soit les « composants » « objet », « interface », « microactionneur », « microcapteur » et « microcontrôleur », qui ont leur propre architecture et configuration.

5.2.6 Le corps d'architecture

Pour modéliser l'aspect distribué de *smart surface*, nous utilisons l'instruction *generate* pour chaque composant MEMS (*gen1*), microsenseur (*gen2*) et microcontroller (*gen3*). Cette instruction très puissante permet l'écriture automatique itérative ou conditionnelle de code. Ici, les variables *i*, *j* et *k* correspondent respectivement aux composants MEMS, microsenseur et microcontroller. Ces trois variables varient dans un intervalle de 1 à DIM.

5.2.6.1 Déclaration de la configuration

Les codes VHDL-AMS correspondants du modèle structurel comportemental *smart surface* sont présentés dans la figure 5.11. La partie **Entity** est identique par rapport au modèle comportemental, uniquement la partie architecture change. Le modèle structuré comprend les composants *air-pressure* (air), *microactionneur* (actu), *microsenseur* (sens), *microcontroller* (cont), *interface* (inter), et *object* (obj), qui sont définis et programmés séparément. Ils appartiennent au *work library* où ils sont appelés dans le programme. Ils peuvent être placés dans n'importe quel ordre.

Nous étudierons dans la section appropriée les résultats de simulation de ce modèle. Abordons à présent l'étude au niveau « modèles composants » de la « smart surface ».

5.3 Modèles composants

Dans cette section, nous étudierons la description au niveau « modèles composants » de la « smart surface ». Il s'agira de décrire l'ensemble des composants qui la composent et qui ont été présentés dans la section précédente, puis de les intégrer dans le « modèle structurel comportemental » avant de simuler le tout.

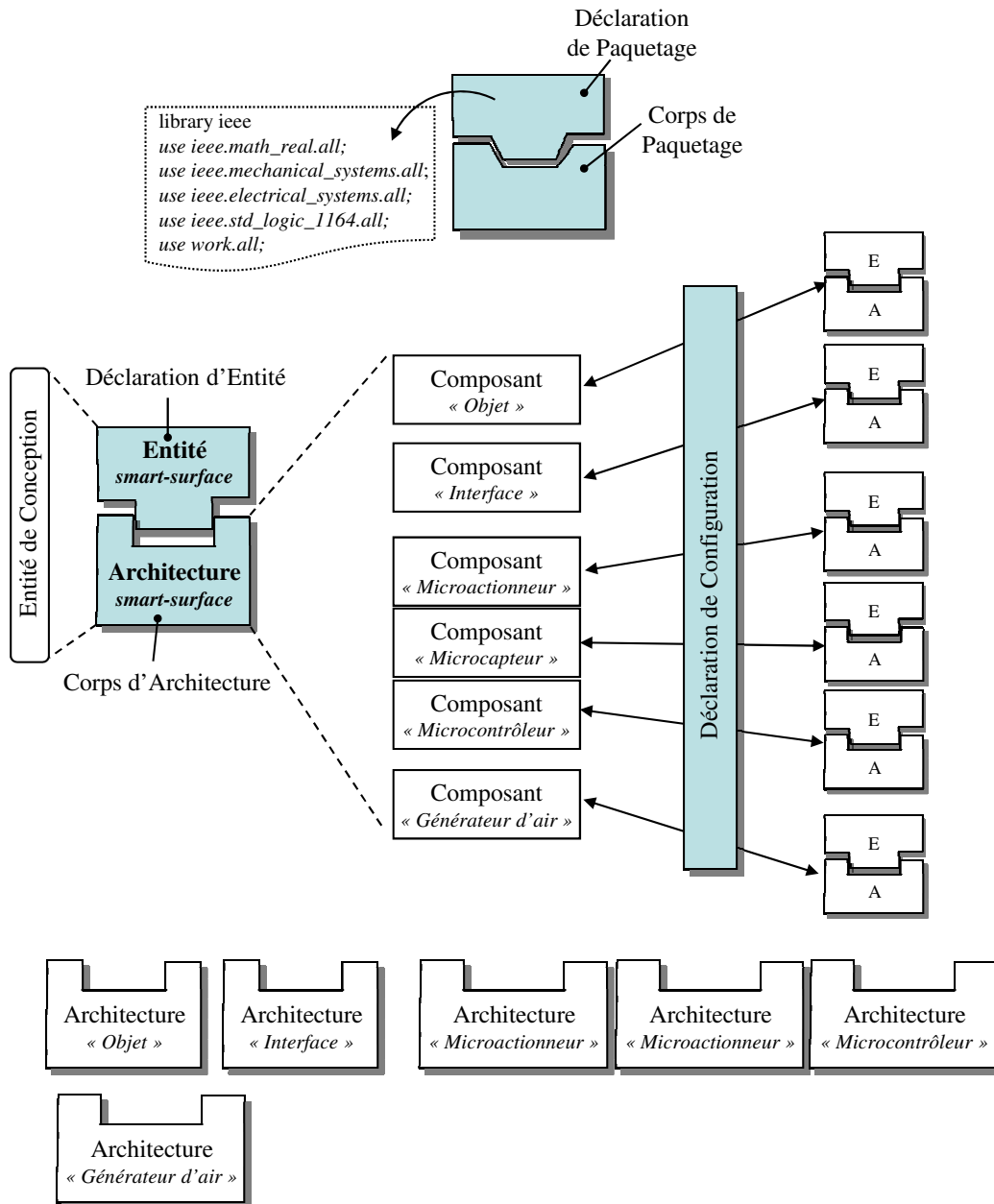


Fig. 5.10 – Structure générale du programme VHDL-AMS du « modèle structurel comportemental » de la « smart surface ».

```

ARCHTECHTURE structural of smart-surface is
.....
BEGIN
--Components declaration
Air:  entity air-pressure generic map(...)
        port map (...);
Gen1: for i in 1 to DIM generate
Actu: entity microactionneur generic map(...)
        port map (...);
        end generate;
Gen2: for j in 1 to DIM generate
Sens: entity microsensor generic map(...)
        port map (...);
        end generate;
Gen3: for k in 1 to DIM generate
Cont: entity microcontroller generic map(...)
        port map (...);
        end generate;
Inter: entity interface generic map(...)
        port map (...);
Obj:  entity object generic map(...)
        port map (...);

END ARCHTECHTURE structural;

```

Fig. 5.11 – Programmation du corps d’architecture du « modèle structurel comportemental » de la « smart surface ».

Pour simplifier notre étude nous nous limiterons dans un premier temps à la description du seul composant « microactionneur » dont nous connaissons la fonctionnalité et la conception à un très haut niveau de détails. Les autres composants pourront alors rester dans la description initialement élaborée dans le « modèle comportemental ».

5.3.1 Modèle « microactionneur pneumatique »

5.3.1.1 Approche

Au niveau « modèles composants », le composant peut être décrit globalement à son plus bas niveau physique mais peut également être décomposé en de multiples « modèles comportementaux » avec différents niveaux de détails [88]. L’ensemble des éléments du microactionneur pneumatique correspondant au composant « microactionneur » sont donnés sur la figure 5.12.

Les dimensions et les microstructures du composant sont construites à partir du masque de conception que nous présentons sur la figure 5.12(a). Les résultats de fabrication sont illustrés par la représentation 3-D du microactionneur de la figure 5.12(b). Le schéma équivalent représentant l’actionnement électrostatique de la micro-valve en face-arrière du microactionneur pneumatique est décrit sur la figure 5.12(c).

Finalement, de manière à descendre jusqu’au niveau de représentation le plus bas, nous détaillerons le profil des microstructures de l’actionnement de la micro-valve et plus spécialement

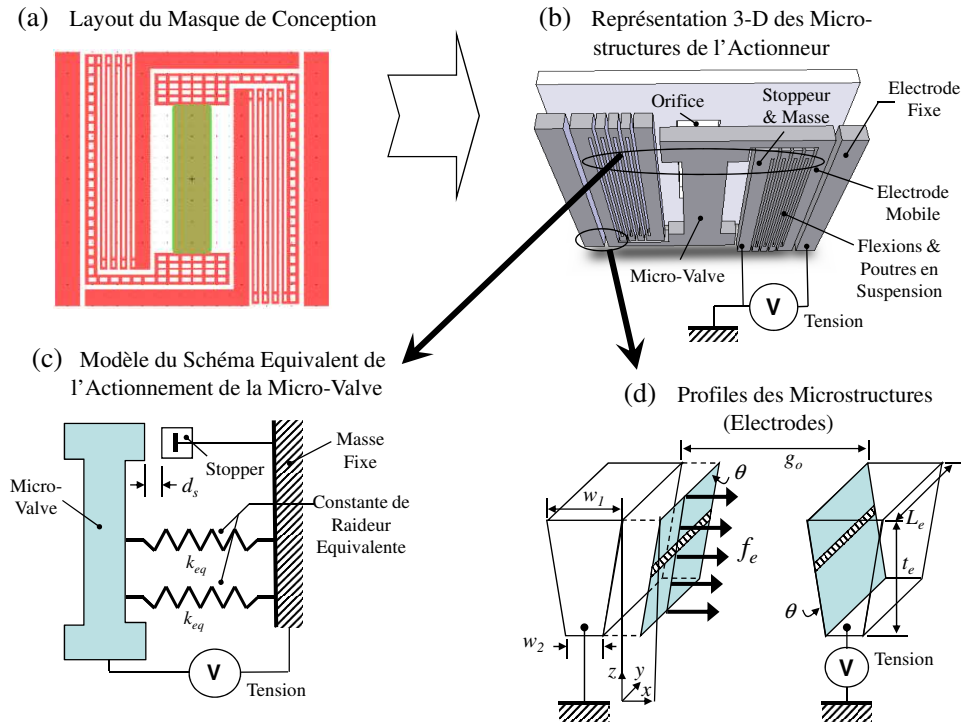


Fig. 5.12 – Description du composant « microactionneur » correspondant aux phases de fabrication et de conception du microactionneur pneumatique.

celui des deux électrodes où se produit l'effet électrostatique, comme le montre la figure 5.12(d). On observe à ce niveau de description la forme « en biseau » des tranches des microstructures qui s'explique par le phénomène de gravure plasma par « Deep Reaction Ion Etching » (DRIE) [89, 90]. On verra ainsi que la force électrostatique ne s'exerce pas avec la même intensité sur l'ensemble des points de la surface latérale des tranches des deux électrodes d'actionnement.

Le modèle du composant « microactionneur » devra donc prendre en compte l'ensemble de ces nouvelles données, s'il veut se situer au plus bas niveau de description physique du système. Nous élaborerons une description du profil des microstructures, illustrées sur la figure 5.12(d) à partir des seules données suivantes :

- Le profil latéral des microstructures dépendant de l'angle (θ) de déclinaison de la gravure DRIE,
- Les largeurs mesurées aux extrémités des électrodes : w_1 et w_2
- La profondeur de gravure des microstructures : t_e
- La longueur des électrodes : L_e
- L'espacement originel (sans défaut de gravure) entre les deux électrodes : g_o

5.3.1.2 Modèle

1. **Calcul de la constante de raideur** : la forme en serpentins des flexions des microstructures d'actionnement est modélisée par une constante de raideur équivalente (k_{eq}), comme

l'illustre la figure 5.12(c). Le mode de calcul de cette grandeur est basé sur les travaux de modélisation des microstructures de microactionneur réalisé par *Fedder* [60]. L'expression de la constante de raideur équivalente prendra alors la forme suivante :

$$k_{eq} = \frac{E(w_1^4 - w_2^4)cot\theta}{56L_e^3} \quad (5.1)$$

E étant le module de Young du silicium.

2. **Modèle du comportement statique** : la force électrostatique (f_e) agissant sur l'actionnement de la micro-valve est produite par une simple structure électrostatique à électrodes parallèles impliquant un champ constant. Cependant, la forme post-fabrication des électrodes parallèles de l'actionneur électrostatique génère un champ électrostatique non-uniforme sur l'espace d'application. Il sera donc nécessaire de calculer la force électrostatique en intégrant sur l'ensemble de la longueur de la tranche des microstructures [60]. La figure 5.12(d) illustre ces conditions de calcul sur les tranches « en biseau » des deux électrodes où s'applique le champ électrostatique. Quand l'électrode mobile se déplace sur l'axe (x), le calcul de la force électrostatique est effectué en intégrant sur l'axe (z), l'expression générale de cette dernière dans le cas classique de deux plaques parallèles. Ainsi, nous obtiendrons la relation de force suivante :

$$f_e = \int_0^{z=t_{st}} \frac{1}{2} \frac{\epsilon_0 L_e V^2}{(g_0 - x + 2z \tan\theta)^2 \cos\theta} dz \quad (5.2)$$

(ϵ_0) étant la constante diélectrique de l'air, (V) la tension appliquée entre les électrodes.

3. **Modèle du comportement dynamique** : l'analyse dynamique sur une dimension du déplacement de l'électrode est établie à partir du modèle de simulation basé sur le comportement de l'actionnement des microstructures. Mécaniquement, la poutre de flexion peut être approximée par une déflexion du système masse-raideur qui est représentée par la grandeur (x). C'est un système type élastique, nous appliquerons donc la relation de la dynamique en rappelant les paramètres du système pour obtenir la relation suivante :

$$f_e = m_s \frac{d^2x}{dt^2} + A \frac{dx}{dt} + 2k_{eq}x \quad (5.3)$$

m_s étant la masse de la microstructure entraînée et A le coefficient de calcul de l'équation du second ordre.

4. **Modèle de la tension de contact (pull-in voltage)** : en reprenant la relation du modèle comportemental dynamique (5.3) et celles des forces de raideur ($-2k_{eq}x$) et électrostatique (f_e), nous allons pouvoir déterminer l'expression de la tension de contact ($V_{pull-in}$) qui s'applique lorsque la distance entre les deux électrodes est au tiers de l'origine ($x = g_0/3$).

On obtient ainsi :

$$V \geq V_{pull-in} = \sqrt{\frac{E(w_1^4 - w_2^4)(2g_0 + 6t_e \tan \theta)g_0^2 \cot \theta \cos \theta}{189\epsilon_0 t_e L_e^4}} \quad (5.4)$$

5.3.2 Description VHDL-AMS

5.3.2.1 Architecture du modèle MEMS

Généralement, le contenu d'un composant peut être un modèle comportemental, ou bien un sous-système constitué de composants interconnectés, qui permettent de construire des modèles hiérarchiques. Dans le cas du code d'un composant « microactionneur », nous développons une *entity* et une *behavioral* architecture, puis les intégrons dans le modèle structurel comportemental *smart surface* développé précédemment. L'architecture est simplement établie en fonction des relations entre les variables de port dans la forme des équations différentielles ordinaires (ODEs) et des équations algébriques (AEs), données dans 5.1, 5.2, 5.3 et 5.4.

Finalement, nousinstancions le composant « microactionneur » avec d'autres composants du modèle structurel « smart surface » et complétons les interconnexions dans le modèle principal. Par exemple, nous interconnectons « microactionneur » et le composant « microcontroller » instancié avec *enable* et les variables $EN(i)$ pour autoriser le contrôle de micro-valve. Les opérations similaires sont effectuées pour direction et les variables $DIR(i)$, *operation* et les variables $OP(i)$ avec le composant interface. Les codes VHDL-AMS correspondants sont présentés dans la figure 5.13.

5.3.2.2 Intégration du modèle « microactionneur » dans le modèle structurel

La figure 5.14 montre l'instanciation du modèle « microactionneur » dans le modèle structurel comportemental. Étant donné que la structure du modèle structurel comportemental est préparée pour l'intégration de différents « composant », cette procédure du modèle « microactionneur » est réalisée tout simplement avec la connexion de ses ports avec les autres composants du modèle.

5.4 Résultats de simulation

5.4.1 Conditions de simulation

Tous les résultats de simulation présentés ci-dessus ont été réalisés sous le logiciel SMASH 7.0 de la société Dolphin Integration® . Nous soulignerons que durant nos travaux de modélisation et de simulation, nous avons été amenés à suivre l'évolution de ce logiciel. En effet, nos développements ont nécessité parfois certaines améliorations des versions successives de SMASH. Nous sommes ainsi passés de la version 5.4 à la version 7.0. en cours de thèse. A noter que de nombreuses modifications du logiciel SMASH ont été soutenues par *Y. Hervé* et la société

```

--Entity
ENTITY MEMS IS
GENERIC (Ms, Rd, Ds, W1, W2, Theta, Eps,
         Eyoung, Le, Te, Go : REAL);
PORT (TERMINALVT: ELECTRICAL);
PORT (enable: in bit; direction: in std_logic_vector (1 downto 0); operation: out bit);
END Microactuator;
.....
--Architecture
ARCHITECTURE behavioral OF MEMS IS
QUANTITY X, V, Fe, Fr, Keq, Vpullin : REAL;
QUANTITY Vin ACROSS VT;
SIGNAL shock1, shock2; BOOLEAN;
BEGIN
Shock1 <= not X'above(-Ds);
Shock2 <= X'above(Ds);
BREAK V => -Rd*V WHEN shock1 or shock2;
X'DOT == V;
IF Vin'above(0.0) use
Fe == 0.5.Eps*Le*Te*Vin**2/((Go-abs(X))*cos(Theta/
180.0*MATH_PI)*(Go-abs(X)+2.0*Te*tan(Theta/
180.0*MATH_PI)));
ELSE
Fe == - 0.5.Eps*Le*Te*Vin**2/((Go-abs(X))*cos(Theta/
180.0*MATH_PI)*(Go-abs(X)+2.0*Te*tan(Theta/
180.0*MATH_PI)));
END use;
BREAK on Vin'above(0.0);
Keq == Eyoung*(W1**4-W2**4)/(56.0*Le**3*
cot(Theta/180.0*MATH_PI));
Fr == 2*Keq*X;
Ms*V'dot == Fe - Fr;
Vpullin == sqrt(8.0*Go**2*(Go+3.0*Te*tan(Theta/ 180.0*
MATH_PI))*Keq/(27.0*Eps*Le*Te));
END ARCHITECTURE behavioral;
.....

```

Fig. 5.13 – Extrait du programme VHDL-AMS du composant « microactionneur »

```

.....
-- Instantiation
Gen1: for i in 1 to DIM generate
Actu: entity MEMS
generic map(Ms => Ms, Rd => Rd, Ds => Ds, W1 => W1,
W2 => W2, Theta => Theta, Eps => Eps, Eyoung => Eyoung,
Le => Le, Te => Te, Go => Go)
port map (enable => EN(i), direction => DIR(i), operation =>
OP(i));
END generate;
.....

```

Fig. 5.14 – Extrait du programme VHDL-AMS de l'instanciation du composant « MEMS »

Systems'ViP avec qui nous sommes en contact étroit et qui a une très grande expertise de développement des outils tiers autour de ce noyau.

Un exemple de problème que nous avons rencontré provenait de la description des éléments distribués de la « smart surface ». En utilisant l'instruction « generate » pour démultiplier notre modèle, nous nous sommes retrouvés face à des contraintes de calcul et d'interprétation du logiciel. Cependant, grâce à la capacité des versions récentes de SMASH pour supporter des jeux d'équations de très grande taille, nous avons réussi à dépasser ces limitations. Ceci était très difficile à réaliser avec les premières versions de travail, voire impossible avec les autres logiciels du marché dont hAMStEr avec lequel nous avons débuté nos recherches.

Un autre problème rencontré lors de nos développements concernait l'interprétation de certaines particularités du langage VHDL-AMS. Lorsque l'on simule le code VHDL-AMS, le simulateur commence toujours par chercher le point de repos statique. A cause de la complexité du modèle, il était souvent impossible pour le simulateur de le trouver. Avec les anciennes versions de SMASH, l'échec de cette opération entraînait l'arrêt de la simulation. Avec la version 7.0 de SMASH ce problème a été résorbé. Par contre, nous avons rencontré quelques perturbations du système qui retardent l'avancement de certains travaux de simulation.

Il a été donc choisi de se fixer sur une seule version de SMASH : la version 7.0.

5.4.2 Modèle « microactionneur pneumatique »

Dans la figure 5.15, nous montrons les résultats de simulation du comportement de la micro-valve électrostatique quand nous appliquons une tension entre -150 V et 150 V pendant 0,03 seconde. Les paramètres que nous avons utilisés sont donnés dans le tableau 5.1.

Paramètres	Description	Valeur	Unité
m_s	Masse de la microstructure	$6,61 \times 10^{-9}$	kg
L_e	Longueur de l'électrode	900×10^{-6}	m
w_1	Largeur du haut de l'électrode	10×10^{-6}	m
w_2	Largeur du bas de l'électrode	$6,5 \times 10^{-6}$	m
t_e	Épaisseur de l'électrode	100×10^{-6}	m
ϵ_o	Constante diélectrique du vide	$8,85 \times 10^{-6}$	F/cm
E	Module de Young	$1,3 \times 10^{11}$	Pa

Tab. 5.1 – Paramètres du modèle du composant MEMS

La position initiale de la micro-valve se trouve au milieu sous l'orifice du microactionneur, sa direction du mouvement dépend du signe de la tension appliquée. Dans cette simulation, nous voulons observer la réponse du comportement de la micro-valve dans les deux directions opposées. C'est pourquoi nous avons appliqué une tension entre -150 V et 150 V.

Dans la figure 5.15, la première courbe représente une tension alternative entre -150V et 150V fournie par le générateur de tension d'alimentation. Quand la tension d'alimentation atteint la tension de contact ($V_{pull-in}$), la valve s'ouvre brutalement. La valve est stoppée ensuite par le stoppeur qui se trouve à $15 \mu m$ de la position centrale de l'électrode mobile. A deux reprises, la tension d'alimentation revient à zéro, sans la force électrostatique, les poutres suspendues tirent la micro-valve vers la position centrale. La micro-valve s'arrête après une oscillation dont la durée dépend fortement de la raideur de ressort des poutres en suspension.

Dans la figure 5.15, nous avons observé l'ouverture et la fermeture brutale de la micro-valve. Ce phénomène de la tension de contact ($V_{pull-in}$) est une caractéristique essentielle du comportement de la valve. Pour notre microactionneur, les tensions de contact ($V_{pull-in}$) varient en fonction de la forme des poutres en suspension.

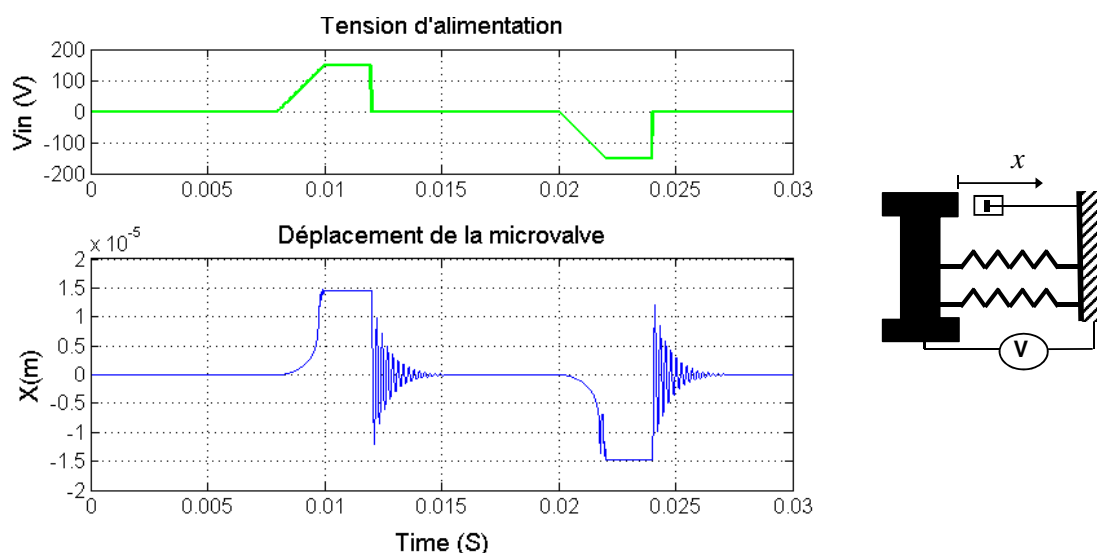


Fig. 5.15 – Résultats de simulation du comportement de la micro-valve

La figure 5.16 présente un exemple de l'étude de la tension de contact ($V_{pull-in}$). Nous appliquons une tension d'alimentation montante pour observer le déplacement de la micro-valve. Nous constatons qu'au fur et à mesure que la tension d'alimentation augmente, la micro-valve s'ouvre de plus en plus. Une fois atteinte une certaine tension, la valve s'ouvre totalement, ce qui est illustré par une montée brutale de la courbe. Selon la théorie que nous avons présentée auparavant, cette tension correspond effectivement à la tension de contact ($V_{pull-in}$). Une étude plus complète sur le changement de la tension de contact ($V_{pull-in}$) en fonction de la forme des poutres en suspension est présentée dans l'Annexe.

5.4.3 Modèle « surface active »

Pour valider le modèle comportement global, plusieurs simulations ont été réalisées. Dans cette section, nous avons procédé un convoyage 1-D de l'objet en utilisant une rangée de cinq

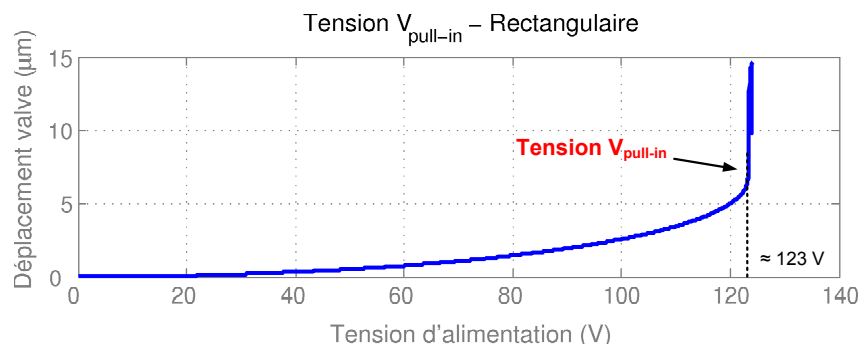


Fig. 5.16 – Résultat de simulation sur la tension de contact ($V_{pull-in}$) - Forme rectangulaire

microactionneurs. Le jet d'air est généré par chaque élément quand l'extrémité de l'objet est détectée dans la zone d'ouverture de l'orifice. Les paramètres utilisés dans le modèle sont présentés dans le tableau 5.2.

Paramètres	Description	Valeur	Unité
m_o	Mass de l'objet	$6,6 \times 10^{-6}$	kg
t_o	Épaisseur de l'objet	$2,5 \times 10^{-6}$	m
w_o	Largeur de l'objet	$4,5 \times 10^{-3}$	m
L_o	Longueur de l'objet	$4,5 \times 10^{-3}$	m
C_{xp}	Coefficient de pression	1,11	
C_{xf}	Coefficient de frottement	0,004	
ρ	Densité de l'air	1,3	kg/m ³

Tab. 5.2 – Paramètres du modèle de la matrice de microactionneurs pneumatiques.

Les résultats de simulation montrés dans la figure 5.17 présentent différentes caractéristiques de l'objet (hauteur, accélération, vitesse) en fonction des conditions fluidiques (vitesse du jet d'air, force de traînée, résistance de l'air).

Les réponses appropriées de la vitesse, l'accélération et la hauteur de l'objet en fonction de la vitesse du jet d'air, la force de traînée et la résistance de l'air ont été obtenues. Quand la partie arrière de l'objet entre dans la zone d'ouverture des micro-valves (1, 2, 3, 4 ou 5), un jet d'air est généré, une force de traînée est appliquée sur cette extrémité de l'objet, qui accélère et la vitesse de l'objet augmente. Après le passage de la cinquième micro-valve, la vitesse de l'objet atteint 0,05m/s.

5.4.4 Modèle « Surface Active » incluant le composant « Microactionneur »

La figure 5.18 montre les résultats de simulation du modèle structurel comportemental incluant le composant MEMS. Ces résultats correspondent bien aux résultats de la section précédente. En même temps, le déplacement des cinq micro-valves a été reproduit comme il est dans le modèle composant. Ces résultats illustrent la capacité du langage VHDL-AMS dans la

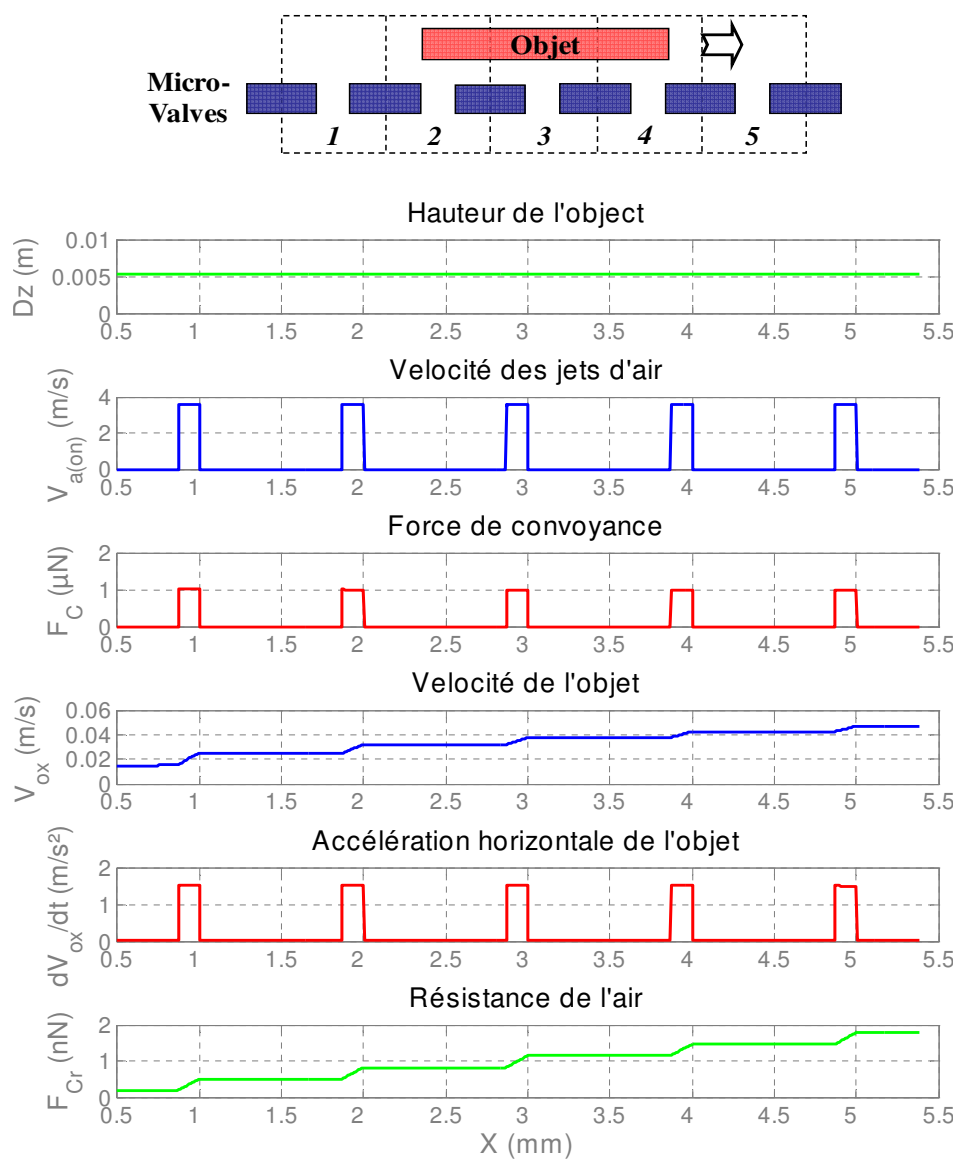


Fig. 5.17 – Résultats de simulation du modèle structurel.

modélisation et la simulation complexe.

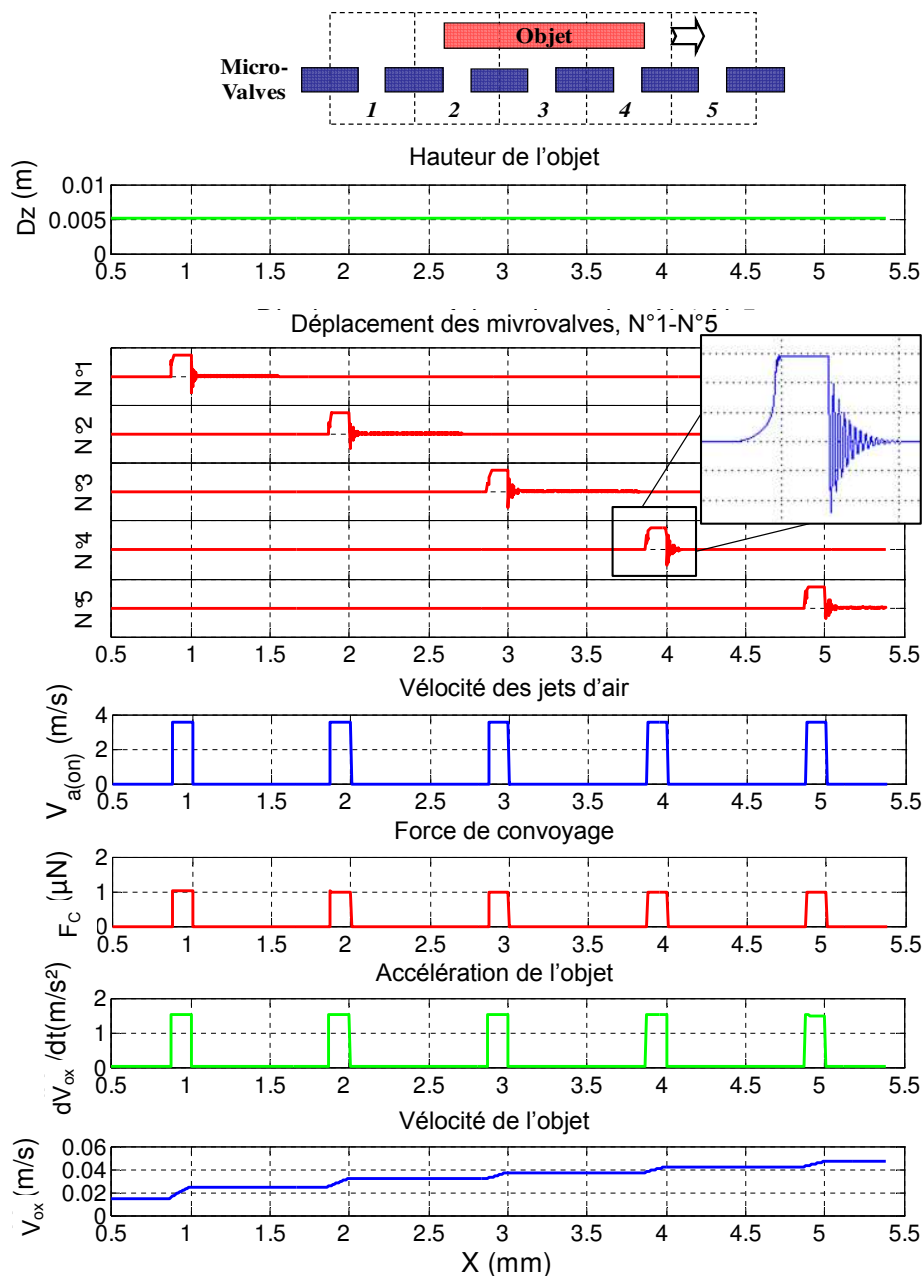


Fig. 5.18 – Résultats de simulation du modèle structurel avec micro-valve

5.5 Conclusion

Dans ce Chapitre 5, nous avons établi avec succès le « modèle structurel comportemental » qui correspond au modèle décrit au deuxième niveau d'abstraction de notre hiérarchie de modélisation.

Dans la première partie, ce modèle structurel comportemental est construit à partir du « modèle comportemental » présenté dans le Chapitre 4 après trois étapes : « identification », « décomposition » et « interconnexion ». Cette deuxième étape dans notre hiérarchie de modélisation a donné une vue « portée de l'intérieur » du système. Le modèle devient un assemblage de sous-modèles qui ont chacun leur propre fonctionnalité. Ce nouveau modèle « structuré » nous permet de gérer séparément tous les membres « composant ».

Dans la deuxième partie, nous avons développé un modèle « microactionneur » qui décrit le comportement physique d'un microactionneur fabriqué à base de technique MEMS.

Finalement, le modèle « microactionneur » est intégré avec succès dans le modèle structurel comportemental. Les résultats de simulation correspondent à nos attentes sur le comportement du modèle global et du « composant » individuel.

La procédure de décomposition et d'intégration de modèle réalisée dans ce chapitre a impliqué le traitement de différents domaines physiques. Cela montre encore une fois la capacité du langage VHDL-AMS dans la modélisation multi-domaines.

Chapitre 6

Méthodologie de validation VHDL-AMS d'un contrôleur décentralisé VHDL

6.1 Objectifs et principes

6.1.1 Objectifs

Les FPGAs appartiennent à la vaste famille des composants logiques programmables. Ceux-ci peuvent être définis ainsi : une matrice comprenant des blocs logiques élémentaires (combinatoires et / ou séquentiels) configurables, reliés entre eux par des interconnexions programmables. Des cellules mémoire contrôlent la partie logique ainsi que les connexions de façon à ce que le composant remplisse l'application souhaitée. Les FPGA les plus denses peuvent atteindre aujourd'hui plus de trente millions de portes logiques élémentaires et peuvent travailler à des fréquences d'horloge système supérieure à 500 *MHz*. Cependant, il convient de souligner que ce type de donnée est extrêmement variable et ne peut représenter qu'une photographie à un instant précis [91].

De par leur structure interne, les FPGAs se prêtent parfaitement à l'élaboration d'architecture de systèmes de contrôle. C'est avec l'objectif de profiter également de la capacité exceptionnelle de ces composants qu'est venue l'idée de conception d'un contrôleur à base de FPGA. En effet, le type matriciel de ces composants peut astucieusement être exploité pour le pilotage localisé de l'ensemble des éléments d'une structure équivalente au dispositif de micromanipulation distribué à base de MEMS étudié dans ce mémoire.

Cependant, une conception « matériel » de ce niveau se doit d'être parfaitement optimisée pour permettre l'établissement d'une architecture adéquate. Il faudrait idéalement que l'architecture du FPGA soit validée à plusieurs niveaux de description dans son contexte fonctionnel. C'est ce que nous entreprendrons dans ce chapitre en appliquant notre flot de modélisation basé

sur les principes du « Prototypage Virtuel Fonctionnel » à la conception envisagée. Abordons à présent cette méthode.

6.1.2 Approche de conception du contrôleur

Rappelons les principes de conception d'un circuit intégré à partir de plusieurs niveaux d'abstraction. Ce type d'approche de conception, qui s'applique aussi bien aux circuits ASIC qu'aux circuits FPGA, s'effectue selon une méthode hiérarchique descendante, appelée aussi conception *top-down* [56]. L'approche descendante part du système en circuits puis sous-circuits et évolue ainsi jusqu'au schéma composé de transistors. La figure 6.1 représente le flot hiérarchique de cette méthode de conception avec les actions et les modèles associés à chaque niveau d'abstraction.

L'approche descendante part du système en circuits puis sous-circuits et évolue ainsi jusqu'au schéma composé de transistors. La figure 6.1 représente le flot hiérarchique de cette méthode de conception avec les actions et les modèles associés à chaque niveau d'abstraction.

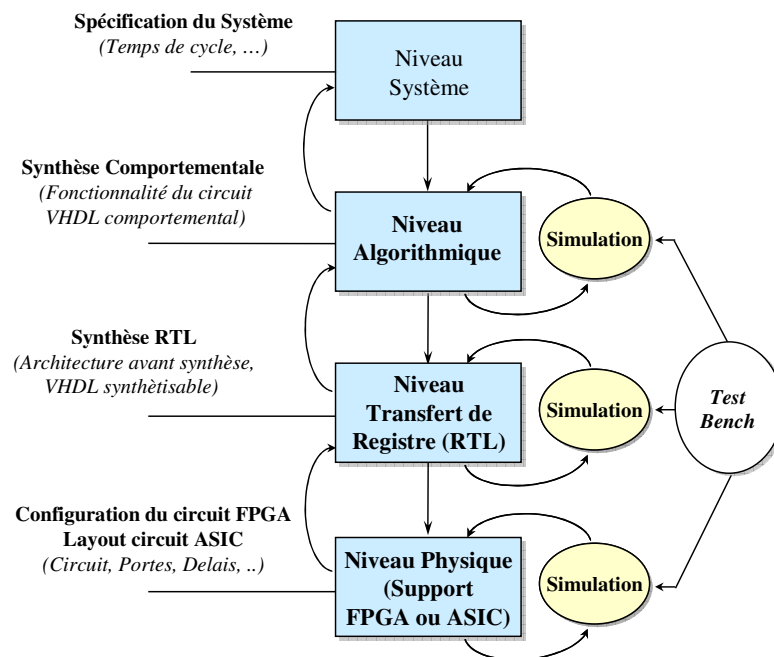


Fig. 6.1 – Schéma hiérarchique de la méthode de conception descendante

Ainsi que le préconise Jerraya [92], les quatre niveaux d'abstraction de la méthode de conception descendante peuvent être définis comme suit :

- le niveau système ou spécification système (*system level*) est le niveau d'abstraction le plus élevé. A ce niveau, aucune architecture et séquençement des opérations ne sont définis.
- le niveau comportemental ou algorithmique. A ce niveau d'abstraction, le circuit est spécifié en termes de pas de calcul, séparés par des points de synchronisation ou des lectures / écritures des entrées / sorties. On parlera de synthèse comportementale dont l'objectif est de découper ces pas en ensembles de cycles d'horloge pour fournir une architecture

synchrone. A ce stade on travaillera en programmation VHDL comportemental.

- le niveau transfert de registre (*RTL : Register Transfert Level*). A ce niveau, les opérateurs sont associés aux modules de librairie, les variables aux points mémoires pour obtenir une représentation en transfert de registre. On étudiera à ce niveau l'architecture de la fonctionnalité et la synthèse RTL, synthèse qui transforme un circuit spécifié pour chaque cycle d'horloge en un ensemble d'équations booléennes. On parlera de programmation VHDL synthétisable.
- le niveau physique est le niveau le plus bas, il prend en considération les informations électriques du système, il possède le plus haut degré de précision dans le modèle. On parlera de configuration du circuit FPGA ou de *layout* pour un circuit ASIC.

6.1.3 Besoin d'un « outil » de validation fonctionnelle

La validation à tous les niveaux de description du modèle VHDL est une étape très importante dans la conception des circuits intégrés. Comme nous l'avons indiqué sur la précédente figure 6.1, à chaque niveau de description VHDL, le circuit modélisé peut être validé par simulation. Cette opération de validation numérique est possible à partir d'un fichier de test que l'on nommera *test bench*. On applique des stimuli sur les entrées d'un modèle VHDL et l'on valide la fonctionnalité de ce dernier à travers les signaux de résultats obtenus.

Cette méthode de validation reste tout de même assez limitée dans le contexte d'une fonctionnalité de commande qui doit être validée dans un environnement analogique et en boucle fermée [93]. En effet, le langage VHDL utilisé pour ces modèles ne permet pas de modéliser les équations différentielles d'un composant analogique tel qu'un CAN, qu'un capteur, qu'un composant de puissance ou même qu'un actionneur électrique. Il faut donc se restreindre à de simples séquences de test de grandeurs analogiques pour espérer valider le modèle du circuit, ce qui peut s'avérer insuffisant ou même erroné.

Pour palier ces lacunes, on peut associer aux modèles VHDL, des modèles HDL de description analogique ou mixte, comme le langage VHDL-AMS, qui nous le rappelons, est une extension normalisée du langage VHDL. L'avantage essentiel de ces langages est de fournir au concepteur un outil de description hiérarchique et de simulation qui s'applique parfaitement à la méthode de conception descendante (*top down*). La validation fonctionnelle à tous les niveaux de description du modèle VHDL est donc possible [94]. La figure 6.2 présente cette approche de validation descendante de conception VHDL à partir d'un environnement de simulation mixte.

Cette approche représente une voie déjà bien connue de validation comportementale de circuits intégrés. Intrinsèquement, elle présente beaucoup de lacunes si l'objectif est une validation de prototype industriel. Seules les approches basées sur les principes du « Prototypage Virtuel Fonctionnel » peuvent répondre à cette ambition. Nous élaborerons ainsi une première méthodologie de validation reprenant les principes de « Prototypage Virtuel Fonctionnel » intégrant nos modèles VHDL-AMS. Suivra une introduction à la stratégie de contrôle décentralisé dé-

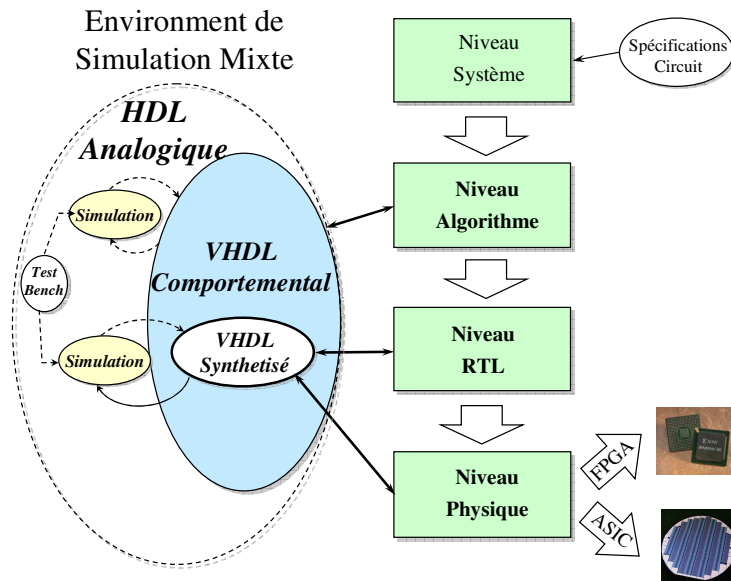


Fig. 6.2 – Environnement de simulation mixte d’une conception descendante VHDL.

veloppés spécifiquement pour cette étude. Puis, nous appliquerons alors la méthodologie à la conception VHDL du contrôleur décrit. Trois étapes de validation sont prévues à chaque niveau de description. Dans une dernière partie, nous présenterons une étude expérimentale sur banc de test du contrôleur intégré sur un prototype FPGA pour relever les cohérences modélisation/expérimentation.

6.2 Méthodologie de validation

6.2.1 Le flot de validation mixte

Il existe dans la littérature plusieurs études à signaux mixtes pour la validation de conception VHDL utilisant VHDL-AMS [95]. Elle confirme que l’on peut valider le comportement d’un circuit intégré numérique ou mixte décrit à plusieurs niveaux d’abstraction. À notre connaissance, il n’existe pas de publication faisant état de validation fonctionnelle au niveau de complexité des modèles que nous proposons. La cause en est la très grande difficulté de simuler des modèles VHDL à un très bas niveau d’abstraction. Il s’agit là d’une limitation due plus aux outils de conception et de simulation existants aujourd’hui sur le marché qu’aux limitations de langage.

En abordant ce travail de méthodologie dans ce domaine de conception, nous ne prétendons pas dépasser ces verrous technologiques mais étendre notre champ d’investigation jusqu’au plus bas niveau d’abstraction possible en appliquant notre approche de modélisation VHDL-AMS. Pour atteindre ce but, nous commencerons par reprendre la méthodologie de « Prototypage Virtuel Fonctionnel » (PVF) de *Y. Hervé* [59], et l’adapter à notre cas de conception.

6.2.1.1 Extraction du « bloc modélisation » du cycle de conception PVF

Le cycle de conception par « Prototypage Virtuel Fonctionnel » peut se décomposer en trois « blocs » de fonction, comme l'illustre la figure 6.3. Le « bloc modélisation » représentant le coeur de la méthodologie, il est celui que l'on souhaite utiliser dans le flot de conception VHDL présenté dans la section précédente.

Très simplement, nous allons extraire ce « bloc modélisation » de son contexte global pour le mixer avec le flot de conception VHDL comme le présente la section suivante.

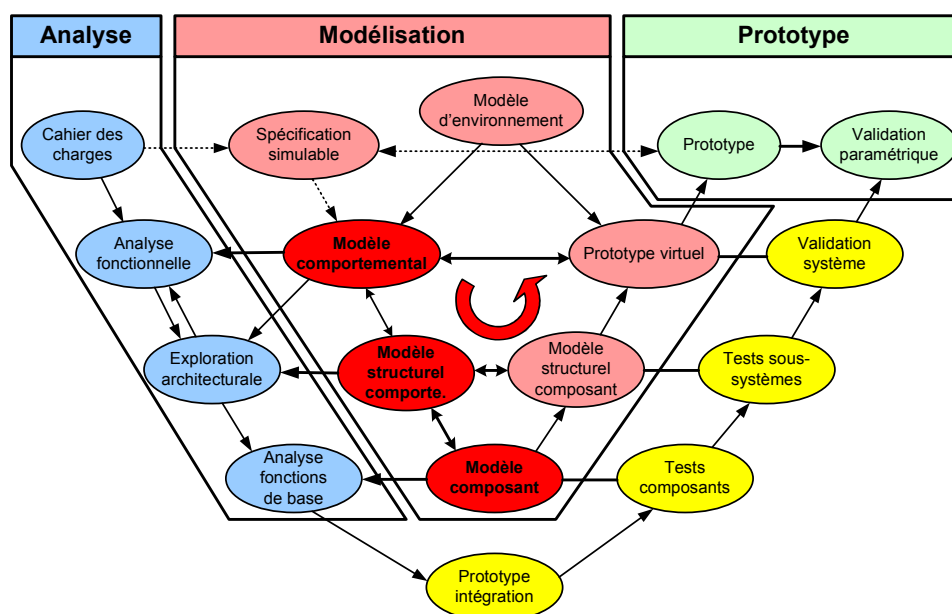


Fig. 6.3 – Classification des « blocs » du cycle de conception PVF

6.2.1.2 Mixage des deux cycles de conception

Le mixage entre les deux cycles de conception se réalise en juxtaposant les mêmes niveaux d'abstraction du flot de conception VHDL et du « bloc modélisation » que nous avons extrait précédemment. Ensuite, nous définirons les actions de validation par simulation « intra-niveau » (simulation au même niveau d'abstraction) et « inter-niveau » (simulation à différents niveaux d'abstraction). Le mixage tel que nous le décrivons est illustré sur la figure 6.4.

6.2.1.3 Étape du flot de validation

On suivra le raisonnement de fonctionnement de la méthodologie en se référant à la description qui en est faite sur la figure 6.4. Très simplement, dans le but de valider un premier flot de conception, nous proposons suivre un flot de validation en six étapes, comme le montre le synopsis de la figure 6.5 :

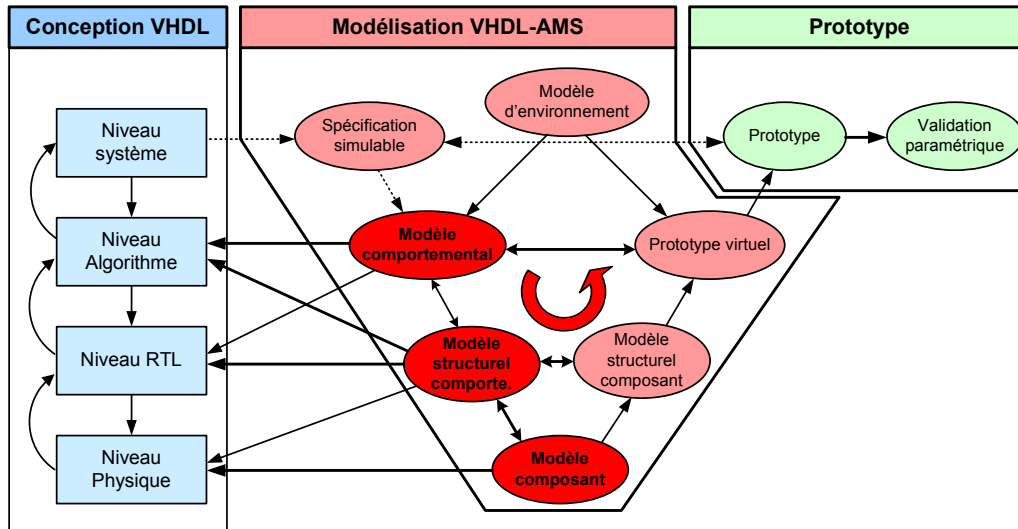


Fig. 6.4 – Mixage des deux cycles de conception.

- **Étape 1** : Si la conception VHDL décrite au « niveau algorithme » est validée dans le « modèle comportemental » on passe à l'**étape 2**.
- **Étape 2** : Si la conception VHDL décrite au « niveau RTL » est validée dans le « modèle comportemental » on passe à l'**étape 3**.
- **Étape 3** : Si la conception VHDL décrite au « niveau RTL » est validée dans le « modèle structurel comportemental » on passe à l'**étape 4**.
- **Étape 4** : Si la conception VHDL décrite au « niveau physique » est validée dans le « modèle structurel comportemental » on passe à l'**étape 5**.
- **Étape 5** : Si la conception VHDL décrite au « niveau physique » est validée dans les « modèles composants » on passe à l'**étape 6**.
- **Étape 6** : Conception validée sur cible FPGA.

Avant d'appliquer ce flot de validation, nous présenterons les principes du contrôleur décentralisé.

6.3 Contrôleur décentralisé

6.3.1 Contrôle distribué : État du domaine

Depuis les années 1990, les chercheurs se sont de plus en plus intéressés aux concepts de systèmes distribués utilisés dans de nombreuses applications technologiques comme la robotique et les structures en réseaux [96, 97]. Ce type de système de contrôle se compose de cellules distribuées, où chaque cellule peut traiter différentes informations. Pour un système complexe et de grande taille, ces structures distribuées permettent de diviser le système en un certain nombre d'entités qui ont leur propre autonomie et capacité de prise de décision. Ainsi, avec la

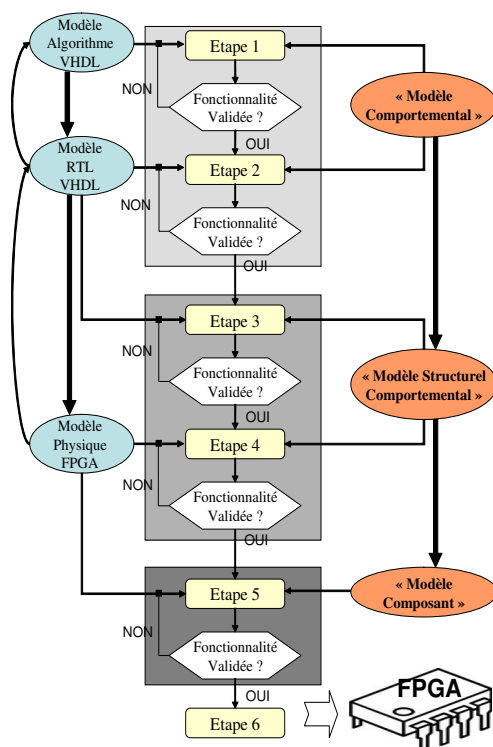


Fig. 6.5 – Flot de validation de la conception à partir des modèles VHDL-AMS

coordination des cellules, le système peut accomplir une mission à grande échelle [98].

Généralement, un système à « contrôle centralisé » consiste en une série de microsystèmes qui se composent uniquement d'un capteur et d'un microactionneur. La stratégie de contrôle du système est effectuée par un contrôleur central. Par conséquent, tous les capteurs et tous les actionneurs doivent être connectés au contrôleur central, ce qui augmente considérablement les connexions des fils. Le coût et la complexité du système augmentent également, alors que la fiabilité du système diminue.

En revanche, dans un système à « contrôle décentralisé », tous les microsystèmes composés de capteur et d'actionneur possèdent aussi leur propre partie intelligente (contrôleur), ils n'ont donc pas besoin de contrôle ou de communication centraux [99]. Dans un système à contrôle décentralisé, la communication et le contrôle se passent localement, les informations s'échangent entre les voisins immédiats. Par conséquent, il n'y a pas de décision globale ou centrale qui est prise pour le contrôle du fonctionnement du système. Pourtant, le système est plus robuste et tolérant en cas de défaillance du contrôle du système. En effet, étant donné que chaque élément de base du système contribue au contrôle du système global, le mauvais fonctionnement d'une partie d'entre eux pourrait ne pas entraîner l'échec total du système, ce qui n'est pas le cas pour le système à contrôle centralisé qui ne possède qu'une unité de contrôle [100].

6.3.1.1 Modèle décentralisé

Dans ce modèle, chaque cellule est considérée comme autonome et possède une partie capteur et intelligente avec une stratégie de décision intégrée. La figure 6.6 présente la matrice des cellules autonomes décentralisées avec l'agrandissement sur la structure d'une cellule et de ses voisines.

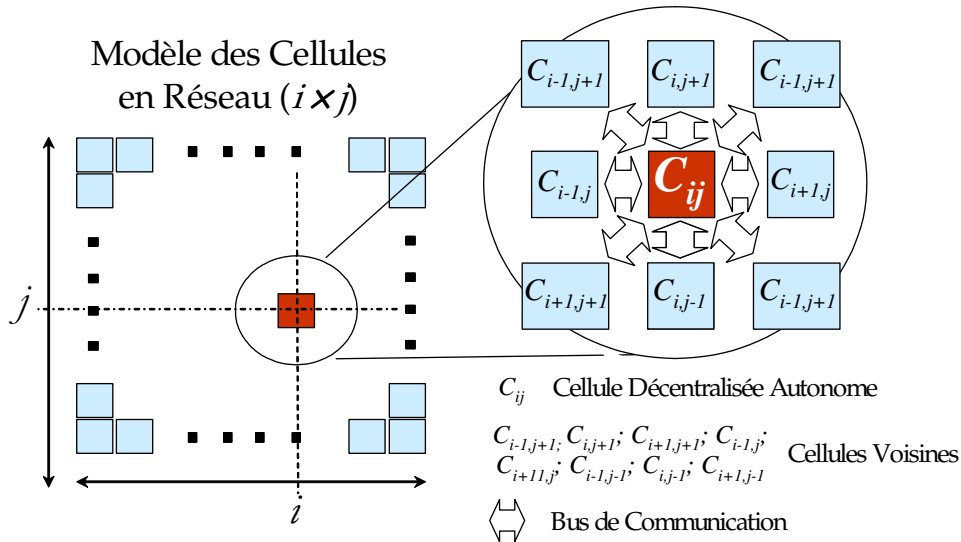


Fig. 6.6 – Architecture du contrôle décentralisé.

Le modèle est représenté par une matrice ($i \times j$) où les indices (i) et (j) indiquent respectivement la position de ligne et de colonne de la cellule, comme l'illustre la figure 6.6. Chaque cellule est représentée par l'indicateur $C_{i,j}$. Le modèle de cellule est basé sur les cinq principes qui peuvent s'appliquer à n'importe quel type de système distribué de micromanipulation :

1. La cellule est supposée avoir son intelligence propre, lui permettant de prendre une décision d'actionnement après interprétation des données captées,
2. La cellule est supposée pouvoir détecter l'objet la couvrant,
3. La cellule peut déplacer l'objet dans les 4 directions transversales qui l'entourent en agissant sur le microactionneur,
4. Chaque cellule de la matrice a le même voisinage que les autres. Pour les cellules qui se trouvent au bord de la matrice, ses voisins sont des cellules de l'autre extrémité du système à convoyage.
5. La cellule est supposée pouvoir communiquer avec ses 8 cellules voisines.

6.3.1.2 Stratégie de contrôle

Nous avons développé et analysé un algorithme basé sur une stratégie de contrôle décentralisé propre à chaque cellule [101]. Cet algorithme sera intégré dans la partie intelligente de la cellule. Nous élaborerons ainsi une méthode simple et intuitive de prise de décision basée sur six règles de

contrôle décentralisé pour répondre à toutes les situations de convoyage du micromanipulateur. Chacune de ces règles est illustrée sur la figure 6.7.

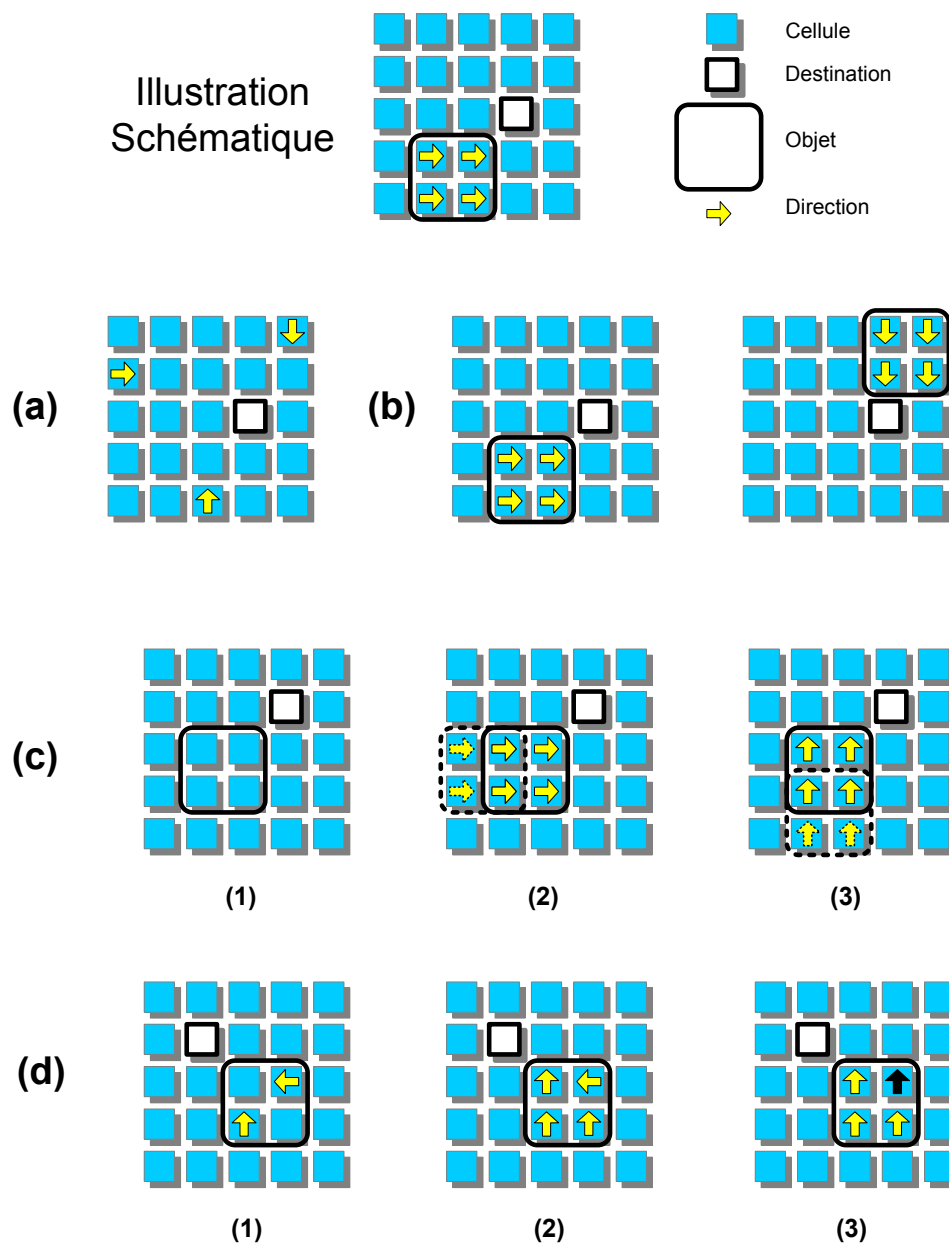


Fig. 6.7 – Illustration des règles de décision des cellules décentralisées lors des phases de micromanipulation d'un objet.

Le détail de toutes ces règles de décision est inscrit dans ce qui suit :

Règle 1 : Seules les cellules couvertes par l'objet sont activées, les autres cellules doivent rester inactives.

Règle 2 : La direction choisie par la cellule doit permettre à l'objet de s'approcher directement de la cellule-destination. La destination désirée de l'objet est toujours représentée

par une cellule du système distribué. L'objet peut être identifié par un carré couvrant quatre cellules. Les directions indiquées par les flèches permettent à l'objet de s'approcher immédiatement de la destination. La figure 6.7(a) illustre cette règle.

Règle 3 : Pour un meilleur contrôle du mouvement de l'objet, tous les cellules actives (sous l'objet) doivent toujours choisir la même direction, quelque soit la position qu'elles occupent par rapport à la destination. La figure 6.7(b) illustre cette règle.

Règle 4 : Quand l'objet peut choisir deux directions pour s'approcher de la destination, la direction précédente sera choisie. La figure 6.7(c) illustre cette règle.

Remarque : En cas de la position initiale de l'objet, une direction par défaut sera donnée.

Règle 5 : En cas de conflit entre l'application de la règle 2 et de la règle 3, nous donnons la priorité à la règle 3. La figure 6.7(d) illustre cette règle.

Remarque : Si les cellules respectent la règle 2, les deux cellules actives doivent choisir deux directions différentes, comme le montre la figure 6.7(d)-1, cela ne respecte donc pas la règle 3. Dans ce cas-là, nous sommes obligés de forcer une de ces deux cellules actives de changer de direction. Quelle cellule doit changer de direction ? La décision des autres cellules est donc décisive. Si elles choisissent la direction nord, comme le montre la figure 6.7(d)-2, la cellule qui a choisi la direction ouest change de direction, comme le montre la figure 6.7(d)-3.

Règle 6 : Une fois que l'objet est arrivé à la destination, les cellules actives doivent être désactivées immédiatement.

Finalement, l'algorithme de prise de décision décentralisé intégré à chaque cellule du système peut être résumé par le synoptique présenté sur la figure 6.8.

La fonctionnalité du contrôleur décentralisé telle que nous l'avons conçu a été simulée et sa fonctionnalité validée hors de son environnement physique.

6.4 Validation Fonctionnelle par Simulation

6.4.1 Adaptation du Modèle VHDL-AMS

Les modèles VHDL de la matrice décentralisée, développés dans la section précédente, s'appliquaient à des structures de microactionneurs pneumatiques quadri-directionnels. C'est-à-dire que chaque cellule était capable de décider et diriger l'objet dans l'une des quatre directions orthogonales au plan. A l'inverse, le composant que nous avons modélisé à l'origine à partir du langage VHDL-AMS est bi-directionnel. Nous ne pouvons donc pas effectuer toutes les opérations de contrôle telles qu'elles ont été conçues. Plutôt que de reprendre la conception du contrôleur, il nous a paru plus opportun de modifier le modèle VHDL-AMS de la « smart surface » pour l'adapter à notre étude. En effet, l'approche de modélisation VHDL-AMS, telle que nous l'avons conçue, doit montrer toute sa flexibilité en s'adaptant efficacement à ce type de transformation

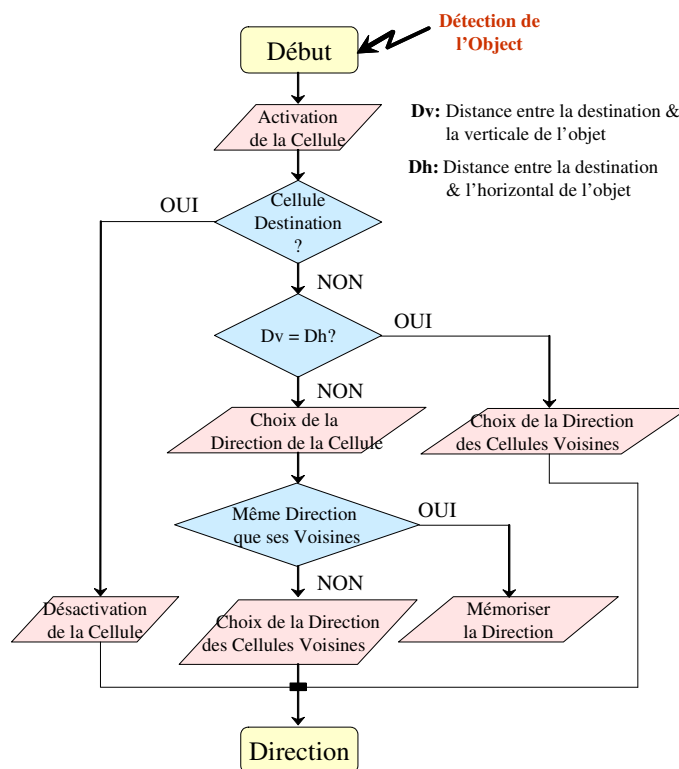


Fig. 6.8 – Synoptique de l’algorithme de prise de décision décentralisé de chaque cellule.

pour cela, il suffit de reprendre le composant « microactionneur » du « modèle structurel comportemental » de la « smart surface » et d’effectuer quelques opérations de description. Cette démarche de modélisation est illustrée sur la figure 6.9.

La représentation de la matrice de microactionneurs pneumatiques à base de micro-valves horizontales et verticales, conçus dans le « modèle structurel comportemental », est reprise sur la figure 6.9(a). L’idée que nous avons eue pour générer un modèle de microactionneurs pneumatiques quadri-directionnels se base sur la décomposition du modèle d’origine en trois étapes :

1. La matrice d’origine de la figure 6.9(a) est dupliquée,
2. La matrice dupliquée est retournée verticalement, comme le montre la figure 6.9(b),
3. Les matrices d’origine et retournée sont superposées de manière à obtenir des cellules équivalentes pour actionnement quadri-directionnel, comme le montre la figure 6.9(c).

La facilité de transformation de notre modèle montre la grande flexibilité de VHDL-AMS. A noter que cette modification peut s’effectuer aussi bien au niveau du « modèle comportemental », que du « modèle structurel comportemental ».

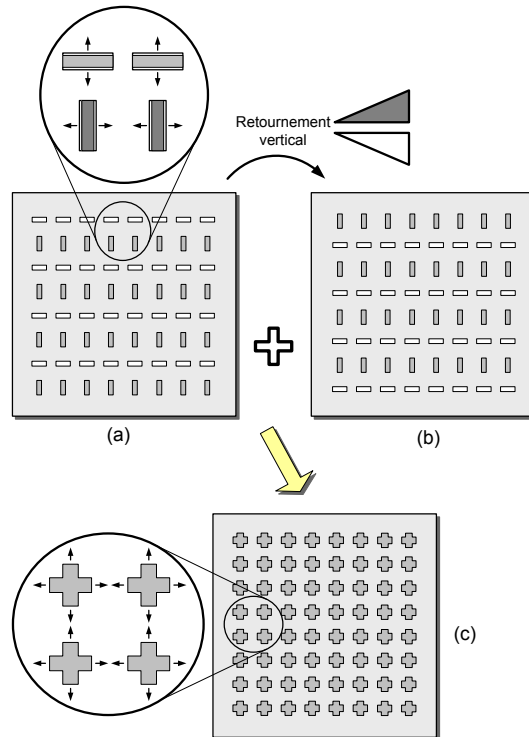


Fig. 6.9 – Transformation du modèle VHDL-AMS bi-directionnel vers un modèle quadri-directionnel des microactionneurs

6.4.2 Niveau Algorithmique

6.4.2.1 Conditions de simulation

Une fois obtenue la nouvelle description du modèle VHDL-AMS, nous pouvons procéder à l'intégration du composant « microcontrôleur » lequel se base sur l'algorithme du contrôleur décentralisé. Le choix du langage de description du composant peut se faire soit en VHDL-AMS, soit en VHDL (comportemental). Pour des raisons de méthode, nous développerons notre contrôle en VHDL-AMS au plus haut niveau d'abstraction de l'algorithme (annoncer les règles). A noter que le « modèle structurel comportemental » comprendra 40×40 cellules intégrant chacune les trois composants « microactionneur », « microcapteur », et « microcontrôleur ». La figure 6.10 illustre cette opération d'implantation.

Nous adopterons un scénario de micromanipulation avec une position initiale et une destination de l'objet qui représentera quatre cellules comme dans la stratégie de contrôle. Nous allons étudier le comportement des cellules de la « smart surface » durant le convoyage de l'objet. Nous adopterons la représentation indiquée sur la figure 6.11 où la destination de l'objet est représentée par un simple point représentant une cellule de la matrice de la « smart surface ».

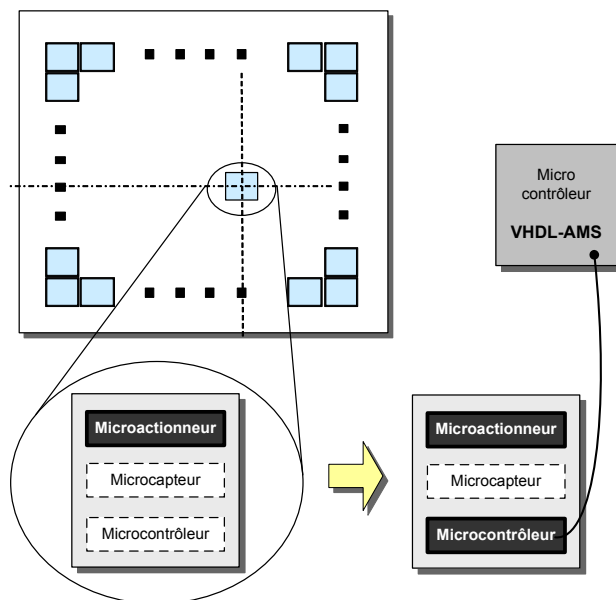


Fig. 6.10 – *Implantation du modèle VHDL-AMS du contrôleur décentralisé dans le structurel de la « smart surface ».*

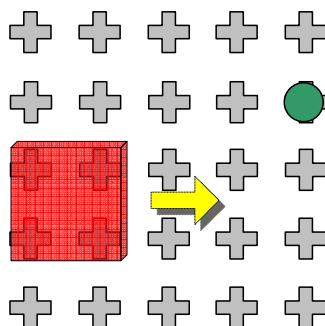


Fig. 6.11 – *Représentation matrice/objet/destination*

6.4.2.2 Simulation « modèle structurel comportemental »

Les résultats de simulation que nous avons obtenus sous la version 7.0 de SMASH à partir des conditions énumérées précédemment sont donnés sur la figure 6.12. On observe de très bons résultats de simulation. Ces derniers ont été collectés puis stockés sous le logiciel Matlab pour établir les graphiques de représentation du parcours de l'objet dans la matrice de cellules. Comme nous n'avons pas simulé de notion d'arrêt de l'objet, on voit ce dernier atteindre le point de destination et continuer son parcours. Puis, de nouveau rechercher son point de destination. Toutes les règles inhérentes à la stratégie de contrôle ont été testées et validées.

Il est aussi très intéressant d'observer les effets fluidiques qui s'exercent sur l'objet. On voit en effet que le déplacement de l'objet présente certaines instabilités de contrôle. On relève par exemple que l'inertie de l'objet entraîne des contraintes sur le contrôle lorsque l'on veut optimiser l'application de forces poussées. A l'inverse, cela prouve encore une fois toute la puissance des modèles développés sous VHDL-AMS qui permettent une description comportementale très précises des phénomènes physiques.

Ces résultats sont les premiers que nous montrons à ce niveau de précision et de complexité du modèle. En effet, les simulations obtenues ne font pas que valider l'intégration d'un composant dans le « modèle structurel comportemental », cette étape ayant déjà été franchie lors du Chapitre 5. Ici, nous avons réussi à valider la coopération de plusieurs cellules lors d'une opération de convoyage d'un objet. Le modèle n'agit donc pas de manière globale comme nous l'avons toujours observé mais de manière distribuée. C'est l'un des objectifs que nous nous étions fixés au début de cette thèse. A notre connaissance, il n'existe pas de résultats avec ce niveau de description dans la littérature du domaine.

A noter que pour palier à ces effets d'instabilité, nous pourrions proposer d'anticiper les mouvements indésirables par un contrôle prédictif. Ainsi, les cellules pourraient agir pour maintenir l'objet avec une trajectoire stable. Notre approche de contrôle permet facilement d'intégrer ce type de stratégie.

En validant la fonctionnalité de l'algorithme de contrôle dans son environnement physique, nous sommes autorisés à passer au niveau inférieur de conception, soit au niveau RTL.

6.4.3 Niveau de Transfert de Registre (RTL)

6.4.3.1 Architecture VHDL du contrôle

La description en VHDL de la matrice de cellules a été réalisée sous Quartus II d'Altera. D'abord, nous proposons de modéliser une matrice de 5×5 cellules en VHDL. La figure 6.13(a) présente le mode de description des entités sous le logiciel Quartus II d'Altera.

Sur la figure 6.13(b) nous présentons l'entité du modèle d'une cellule. La définition des grandeurs d'entrées et de sorties est reprise ici :

- X-Pos et Y-Pos : les coordonnées sur 3 bits de la destination données par un superviseur.

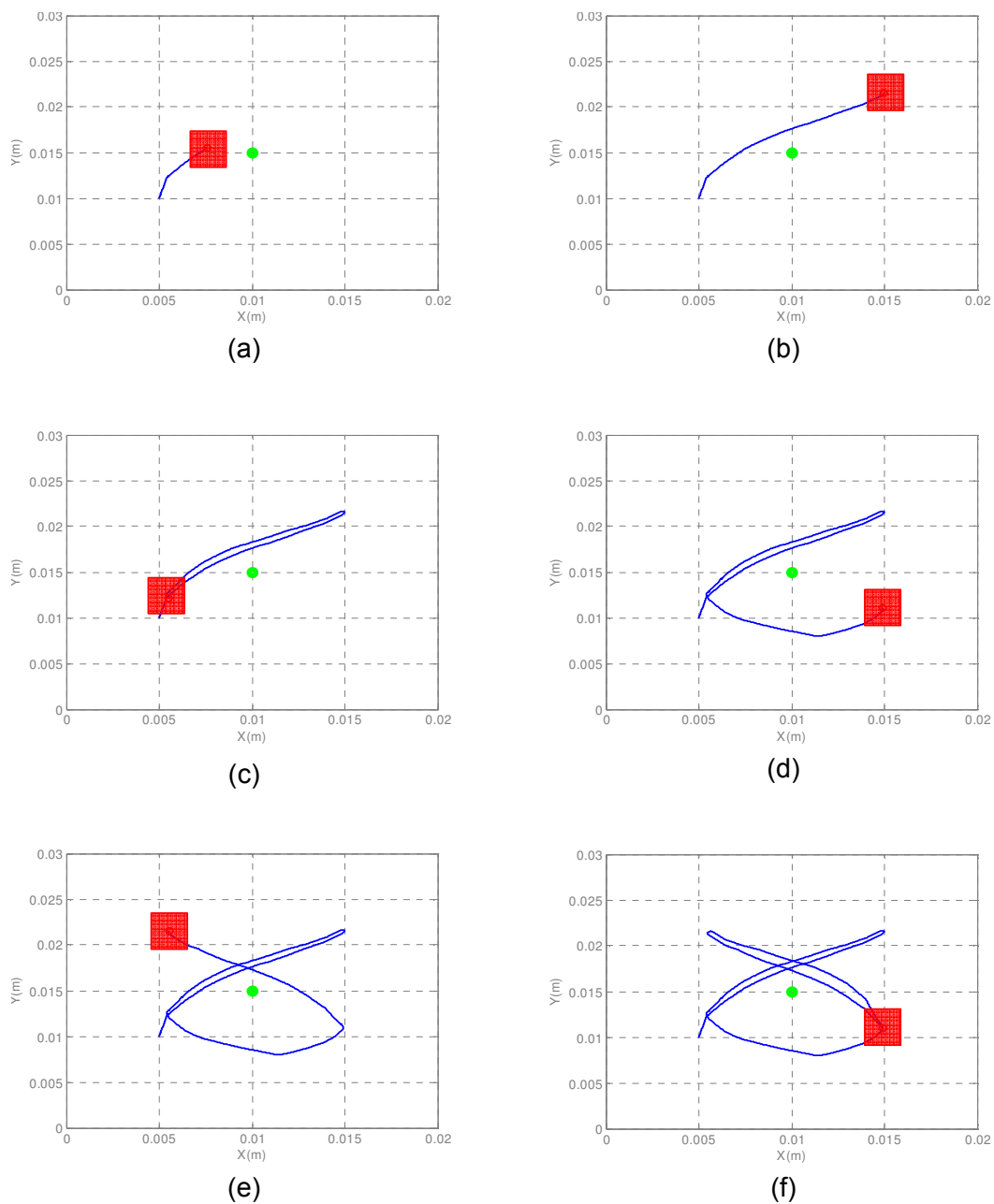


Fig. 6.12 – Simulation du contrôleur décentralisé décrit au niveau algorithmique.

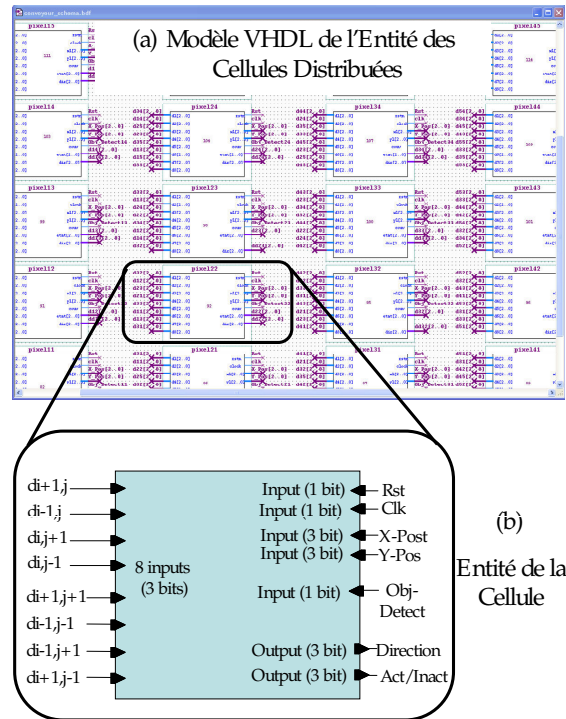


Fig. 6.13 – Représentation de l'entité de la matrice de cellules à contrôle décentralisé. (a) Modèle de l'entité de 5×5 cellules développé sous Quartus II. (b) Entité d'une cellule.

- Clk : l'horloge du circuit qui sera aussi utilisée pour la synchronisation des cellules.
- Act/Inact : l'état de la cellule (activée ou inactivée).
- Les informations concernent les directions choisies en provenance des huit cellules voisine : $d_{i+1,j}$; $d_{i-1,j}$; $d_{i,j+1}$; $d_{i,j-1}$; $d_{i+1,j+1}$; $d_{i-1,j-1}$; $d_{i-1,j+1}$; $d_{i+1,j-1}$.

La méthode décentralisée de prise de décision est basée sur deux blocs principaux, il s'agit du bloc « calcul de la direction », et du bloc « règles de décision ». Le bloc « calcul de la direction » n'exige pas des opérations significatives de calcul, et fonctionne avec un simple opérateur de comparaison.

En ce qui concerne le bloc « règles de décision », nous devons y mettre en oeuvre un processus de contrôle. Généralement, pour réduire au minimum la quantité de ressource de matériel utilisée dans un circuit intégré comme le FPGA, nous utiliserons un processus de contrôle pour réduire la quantité de calcul instantané que nous devons faire à un moment donné. Afin de réaliser le contrôle d'une manière propre, la machine d'états est très souvent sollicitée.

Finalement, nous établirons une machine d'états de décision pour gérer toutes les règles de décision de la méthode décentralisée. Un troisième bloc « gestion de conflit », basé sur une autre machine d'état, a été intégré dans l'architecture afin de gérer le conflit possible entre les décisions prises.

La figure 6.14 décrit l'architecture du modèle VHDL de cellule. Nous retrouvons les trois

blocs « calcul de la direction » (bloc 1), « règles de décision » (bloc 2) et « gestion de conflit » (bloc 3).

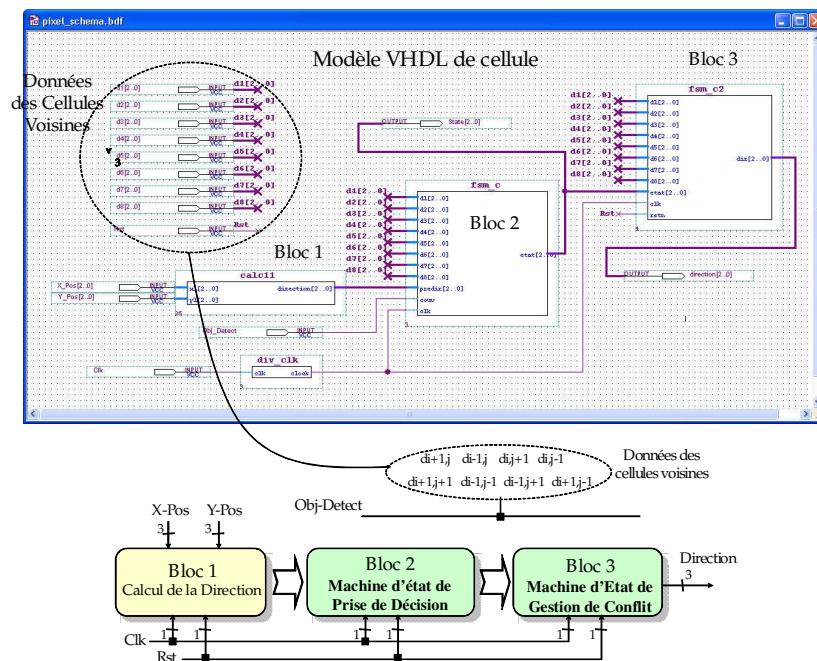


Fig. 6.14 – Architecture du modèle VHDL de la cellule

Le détail fonctionnel des trois blocs du modèle VHDL est donné ici :

- **Bloc 1 : Calcul de la direction**

Ici, nous calculons la direction choisie par la cellule pour convoier le micro-objet vers la destination dont la position est indiquée par les signaux d'entrées ($X - Pos$, $Y - Pos$). Les données de position de la destination sont comparées aux données de position de la cellule elle-même. Selon le résultat de la comparaison, nous pouvons donner la direction à prendre. Seulement les opérateurs de base sont nécessaires à l'élaboration de l'architecture de ce bloc.

- **Bloc 2 : Règles de décision**

Il est possible que le bloc « calcul de la direction » ne puisse pas prendre la bonne décision dans les cas particuliers. Ici, nous établissons toutes les règles de décision en utilisant une machine d'état avec pour objectif de raffiner le choix de direction. La machine d'états développe un système dynamique discret traduisant des vecteurs d'entrée en des vecteurs de sorties qui sera la direction choisie. Les vecteurs d'entrée du modèle sont : les données de cellules voisines, la détection du micro-objet et la direction choisie par le bloc 1. Nous utilisons différents états pour enregistrer l'ancienne entrée, de sorte que quand la prochaine entrée est arrivée une transition puisse être prise sur l'information ancienne. Nous pouvons appliquer ainsi toutes les règles de décision. La machine d'état a été mise en application en utilisant un multi-contexte (*microcode*).

- **Bloc 3 : Gestion de conflit**

Ici, nous établissons un bloc de gestion de conflit. Quand la direction calculée n'est pas logique avec d'autres cellules, une deuxième machine d'état est responsable pour gérer et résoudre le conflit. Les vecteurs d'entrée se composent des données de cellules voisines et des données de direction. La machine d'état a été mise en application en utilisant un multi-contexte (microcode).

L'architecture VHDL a pu être synthétisée et sa fonctionnalité simulée. Les résultats se rapportant à ces travaux ont déjà été reportés [101].

6.4.3.2 Conditions de simulation

On retrouve à ce niveau les mêmes conditions de simulation de la « smart surface » que celle décrite lors de la validation au niveau « algorithme ». Par contre, à présent le contrôleur décentralisé est entièrement décrit en VHDL synthétisable et à l'image de l'architecture VHDL présenté. Nous nous proposons d'intégrer cette architecture telle que nous l'avons conçue dans le modèle VHDL-AMS, comme l'illustre la figure 6.15.

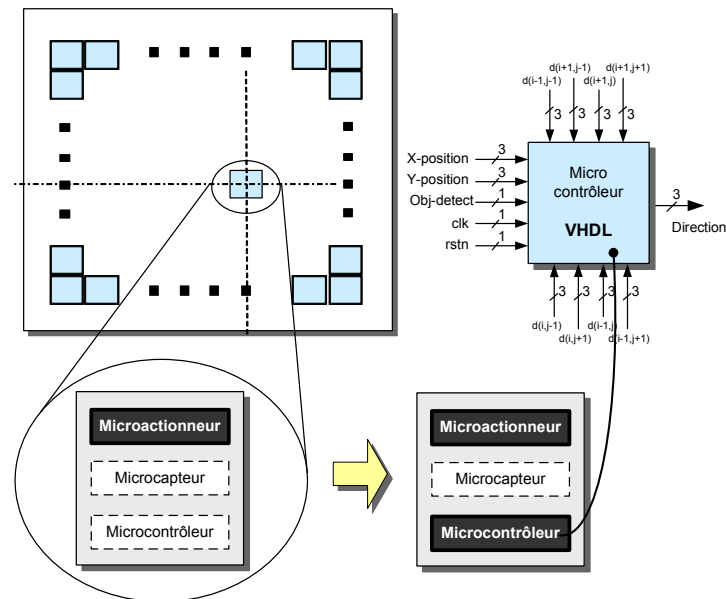


Fig. 6.15 – Implantation du modèle VHDL du contrôleur décentralisé dans le modèle structural de la « smart surface ».

Comme nous l'avons déjà introduit, le langage VHDL-AMS est un standard IEEE (IEEE 1076.1- 1999). Il a été développé comme une extension du langage VHDL pour permettre la modélisation et la simulation de circuits et de systèmes analogiques et mixtes logiques-analogiques. VHDL-AMS constitue un sur-ensemble de VHDL, ce qui signifie principalement que toute description VHDL légale l'est aussi en VHDL-AMS et produit les mêmes résultats de simulation. De même, les extensions apportées dans VHDL-AMS conservent les principes VHDL : modularité, déclarations avant usage, typage fort des données, flexibilité, extensibilité. Ces principes concernent à la fois la manière dont le langage est défini et la manière dont un modèle est écrit.

Pour réaliser l'opération d'intégration du contrôle, nous avons réalisé l'interfaçage VHDL/VHDL-AMS en décrivant les opérations d'instanciation en respectant le code descriptif de la figure 6.16.

```

.....
gen_con1 : for i in 1 to DIM generate
gen_con2 : for j in 1 to DIM generate

    control : microcontroler generic map (x=>i,y=>j,des_x=>des_x,des_y=>des_y)
    port map (dir(i+1)(j),dir(i-1)(j),dir(i)(j+1),dir(i)(j-1),dir(i+1)(j+1),
    dir(i-1)(j-1),dir(i-1)(j+1),dir(i+1)(j-1),position_x,position_y,n1,n2,
    n1_h,n2_h,st1,st2,st1_h,st2_h,arr(i)(j),en(i)(j),dir(i)(j));

end generate;
end generate;
.....

```

Fig. 6.16 – *Extrait du programme d'interfaçage VHDL/VHDL-AMS*

6.4.3.3 Simulation « modèle structurel comportemental »

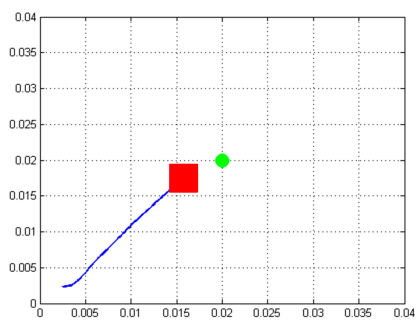
Dans la section précédente, nous avons montré les résultats de simulation réalisés avec le modèle comportemental dont le contrôleur est décrit au plus haut niveau d'abstraction - niveau algorithmique.

Après avoir validé notre contrôleur décentralisé au niveau algorithmique, nous procédons la validation du contrôleur décentralisé au niveau RTL par simulation. La figure 6.17 montre les résultats de simulation que nous avons obtenus toujours sous la version 7.0 de SMASH. Dans ce modèle, nous avons remplacé le contrôleur décentralisé au plus haut niveau décrit en VHDL-AMS par le contrôleur au niveau RTL décrit en VHDL.

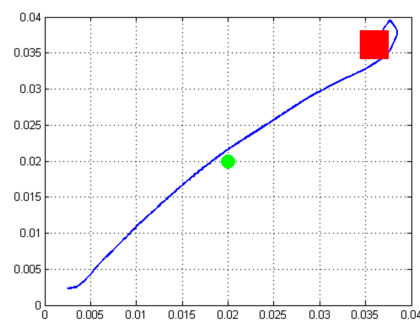
Nous avons pu constater le même rapprochement de l'objet vers la destination tout au long de la course de l'objet par rapport à la validation précédente. L'observation des effets fluidiques qui s'exercent sur l'objet est aussi intéressante. Étant donné que le contrôleur ne contrôle que le rapprochement, mais pas d'arrêt de l'objet sur la destination. Même si l'objet passe tout près de la destination, l'inertie de l'objet l'oblige à continuer son chemin avant d'être changé de direction plus tard.

La similitude entre la simulation au niveau algorithmique et la simulation au niveau RTL correspond à notre attente. Car le changement de niveau d'abstraction ne doit pas modifier la fonctionnalité du contrôleur.

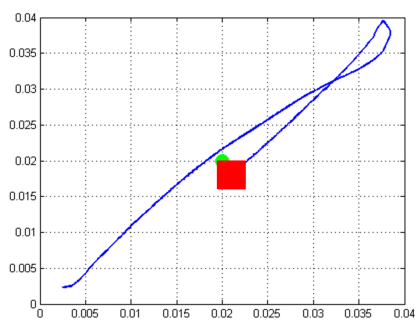
Ce qui nous intéresse dans cette simulation est la communication et la coopération des cellules. Comme il y a de très nombreuses cellules (nous avons simulé une Smart Surface de 40×40 de cellules.) Nous ne pouvons qu'observer cette communication et coopération en suivant quelques cellules. La figure 6.18 est une zoom de la fenêtre d'affichage de résultat de SMASH. Comme les noms des signaux (à gauche de la fenêtre) indique, on affiche ici la direction choisie



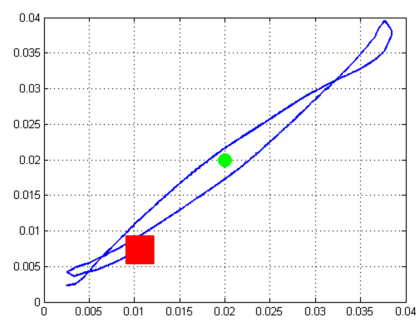
(a)



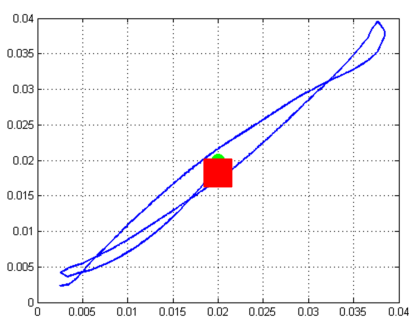
(b)



(c)



(d)



(e)

Fig. 6.17 – Simulation du contrôleur décentralisé décrit au niveau RTL.

par les contrôleurs locaux de quelques cellules. La valeur du signal correspond à la direction que la cellule a décidé de prendre.

Vers 210ms, l'objet arrive sur les cellules [3][3],[3][4],[4][3],[4][4]...(les numéros indiquent respectivement la ligne et la colonne de la cellule.) La décision a été prise immédiatement par toutes les cellules concernés, en l'occurrence, c'est (011), après avoir communiqué avec ses voisins, une partie des cellules concernés ont décidé de changer de direction (001) ou de s'arrêter (000).

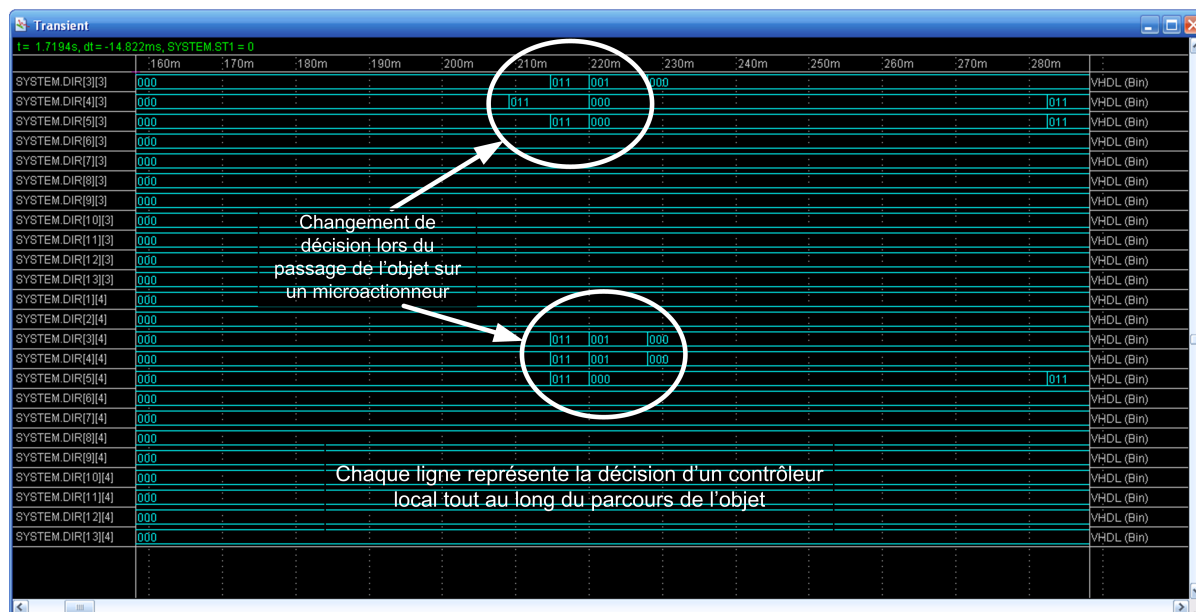


Fig. 6.18 – *Changement de décision des cellules en fonction des décisions du voisinage*

6.4.4 Niveau Physique (ou porte)

Malgré les difficultés rencontrées dans la section précédente, nous poursuivrons ici notre étude de manière à achever le flot de conception au niveau physique (ou porte).

6.4.4.1 Synthèse et GATE LEVEL

Nous présentons ici les résultats d'intégration que nous avons obtenus après la configuration du modèle VHDL RTL de l'architecture du contrôleur sur la cible FPGA.

Nous utiliserons une carte de développement d'Altera (Excalibur Development Kit) équipée d'un FPGA type APEX EP20K200EFC484-2X d'Altera. Cette carte de développement fonctionne avec l'horloge interne de 33 MHz et a une capacité de 8320 unités logiques (LUTs), équivalente à plus de 200.000 portes logiques. La synthèse et l'implémentation du code VHDL ont été effectuées sous Quartus II Altera. Le tableau 6.1 présente les données d'utilisation des ressources du FPGA affichées par le compilateur. Le résultat montre que la conception utilise peu de ressource puisqu'une cellule occupe seulement 1.9% d'éléments logiques de la capacité du

FPGA. En cas de la configuration de 5×5 cellules, malgré un grand nombre de blocs communs entre les cellules, 14.3% d'éléments logiques ont été utilisés, voir le tableau 6.2.

Caractéristiques	Données
FPGA Device	Altera, EP20K200EFC484-2X
Total logic elements	8,320
Part logic elements used	160 (1.9%)
Part Pin used	39/376 (10%)

Tab. 6.1 – Résultats d'implémentation sur une cible FPGA d'une cellule

Caractéristiques	Données
FPGA Device	Altera, EP20K200EFC484-2X
Total logic elements	8,320
Part logic elements used	1,191 (14.3%)
Part Pin used	110/376 (29%)

Tab. 6.2 – Résultats d'implémentation sur une cible FPGA d'un réseau de 5×5 cellule

6.5 Vérification expérimentale

De nouveau nous anticipons sur les résultats de la méthodologie de conception développée ici. L'intérêt étant de montrer la fonctionnalité « réelle » (à l'instar de « virtuelle ») du prototype FPGA que nous avons développé. Toute la démarche expérimentale est présentée dans cette section jusqu'aux résultats de micromanipulation. Une comparaison avec les simulations obtenues au niveau algorithmique sera également étudiée pour valider la cohérence de contrôle des deux approches.

6.5.1 FPGA et plateforme

Ce travail de validation expérimentale a été réalisé en collaboration étroite avec *Y.-A. Chapuis* (LIMMS/CNRS-IIS) et le laboratoire du *Pr. Fujita* (Fujita Lab.) de l'IIS/Université de Tokyo (Japon) [37]. Si la phase d'intégration a pu se développer à l'InESS, les tests sur la plateforme expérimentale ont été effectués au sein du Fujita Lab. au Japon.

6.5.1.1 Implantation conjointe

Structure d'implantation

La figure 6.19 illustre la structure d'implantation du contrôle distribué et autonome de la matrice de microactionneurs pneumatiques. Elle se compose d'une cible « matérielle » type

FPGA où seront exécutés les ordres de décision des cellules et d'une cible « logicielle » où seront traitées toutes les données de perception et de localisation (les éléments capteurs n'étant pas physiquement intégrés sur le composant MEMS). Les séquences de contrôle en boucle fermée peuvent se décomposer de la manière suivante :

1. La caméra capte les images, puis les transfère vers le PC (OS : FREE BSD) ;
2. Les séquences vidéo sont extraites et stockées ;
3. Un algorithme de reconnaissance de forme et de localisation traite les images, puis génère le tableau de donnée contenant les informations de la position de l'objet ;
4. Le résultat est transféré à travers le port USB vers la carte FPGA ;
5. A partir de ce résultat, l'algorithme de contrôle implémenté sur le FPGA génère la décision destinée au microconvoyeur pour déplacer l'objet.

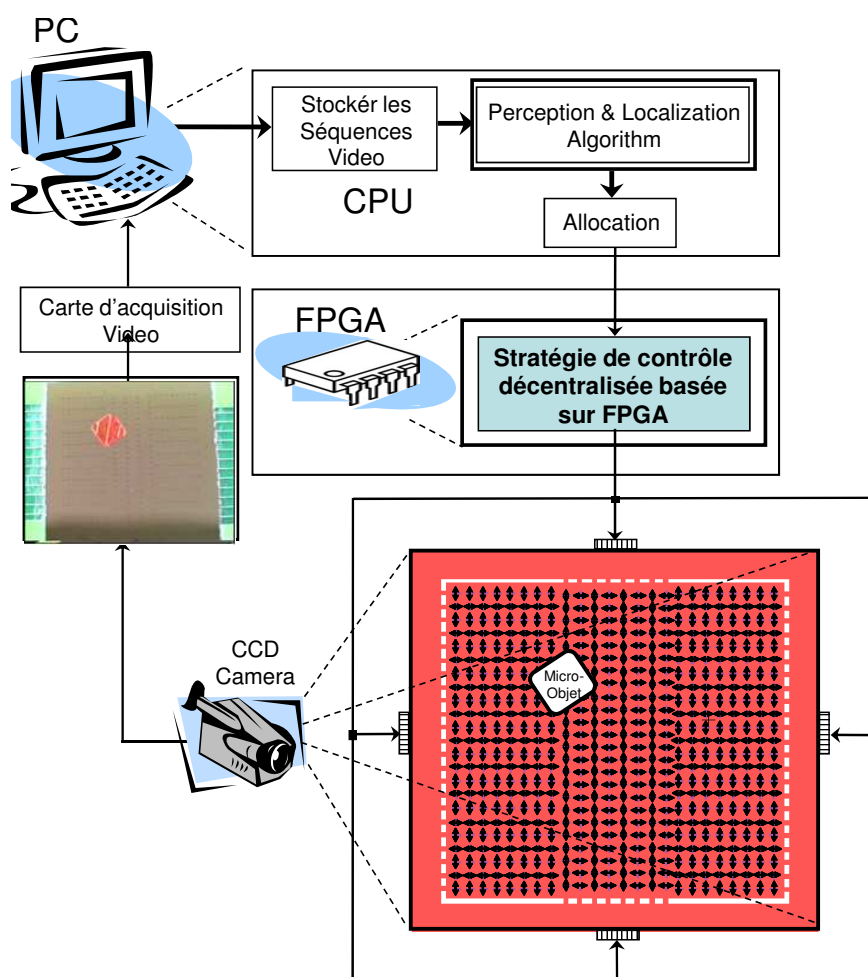


Fig. 6.19 – Structure de contrôle du micromanipulateur distribué à base de MEMS

Algorithme de traitement centralisé

Le système est supposé être autonome. Ce qui signifie que chaque microactionneur devrait pouvoir détecter si un objet la couvre ou pas. Pour rendre ces opérations possibles, nous pro-

posons de générer une image codée de la surface distribuée où chaque pixel sera affecté à un microactionneur de la matrice. Nous avons ainsi une base de données à deux dimensions dont chaque élément est codé soit par « 1 », soit par « 0 ». Concrètement, quand un microactionneur pneumatique est couvert par un objet, elle est codée par « 1 », et dans le cas contraire, elle est codée par « 0 ».

Pour se faire, une caméra CCD avec une résolution de 640×480 (Vainqueur-JVC, GR-DVP7) capture des séquences vidéo de la surface active distribuée avec l'objet en mouvement par l'intermédiaire d'une carte vidéo (I-O DATA, GV-BCTV5/PCI). Toutes les séquences vidéo sont enregistrées dans un fichier en format Device Independent Bitmap (DIB), puis sont traitées par des algorithmes de traitement d'image écrits en langage C++.

6.5.1.2 Interface avec le FPGA

Interface PC/FPGA

Toutes les communications entre le PC et le FPGA se font à l'intermédiaire d'USBMOD4, un module de développement « plug and play » (ELEXOL), qui possède un chip de 8 bits parallèle FIFO pour le port USB (FTDI FT245BM). Ce module USBMOD4 permet au PC de transférer des données à un périphérique avec une vitesse jusqu'à 8 *Mbps*. En utilisant les drivers « virtual COM » du *chip FTDI*, nous pouvons manipuler le port USB comme un port série standard.

La réalisation de la communication entre le PC et le FPGA exige un programme de contrôle qui gère le transfert des données en fonctions de différents signaux de contrôle du *chip FTDI*. Ce programme de contrôle a été écrit en VHDL, puis implémenté préalablement sur le composant FPGA.

Interface FPGA/Drivers

Comme introduit dans le chapitre 2, le dispositif du micromanipulateur à base de MEMS contient 560 microactionneurs commandés par rangées ou par colonne, ce qui réduit les connexions à 128. Malgré le fait que nous n'ayons pas une structure décentralisée, tous nos contrôles s'effectueront de manière parfaitement décentralisée.

Les signaux de contrôle à 8 bits émis par le FPGA peuvent être envoyés vers le micro-système, par l'intermédiaire d'une carte « drivers », contenant 4 32-bit 120 V-driver ICs. (Toshiba, TD62981P). Tous les drivers ICs sont intégrés dans un prototype de carte d'interface interne unique leur fournissant une tension d'alimentation fixée au maximum de 120 Volts DC. Chaque carte d'interface interne peut être communiquée, par l'intermédiaire d'un programme spécifique en VHDL. Ainsi, les données de contrôle sont multiplexées et envoyées en parallèle. La figure 6.20 présente l'architecture du système de contrôle temps-réel de la matrice de microactionneurs, y compris la caméra CCD, le PC, l'interface PC/FPGA, et les drivers entre le FPGA/microsystème.

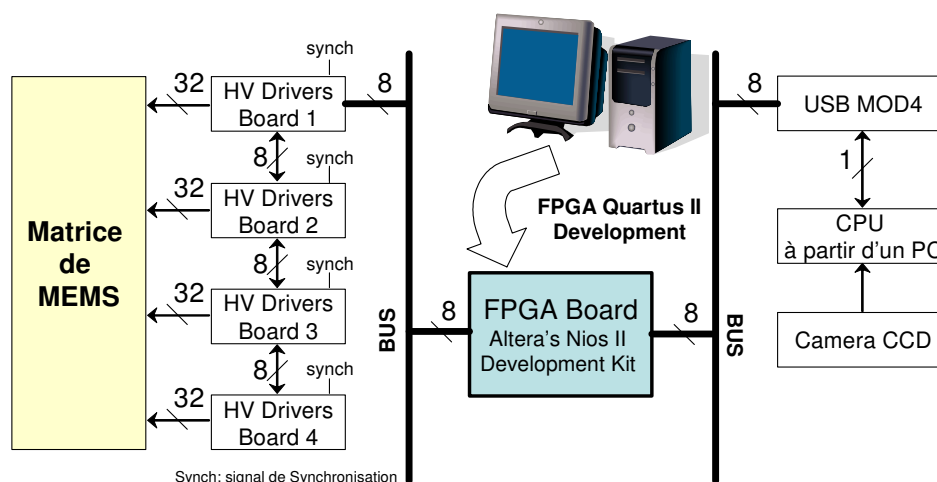


Fig. 6.20 – Architecture du système de contrôle « temps-réel » de la matrice de MEMS

6.5.1.3 Résultats de contrôle

Banc de validation

La figure 6.21 présente l'installation complète du banc utilisé pour la validation expérimentale du système de micromanipulation distribué. Cette installation se compose d'un système à air comprimé, d'une l'alimentation de tension et d'un traitement de calcul pour élaborer la commande à l'aide d'un PC.

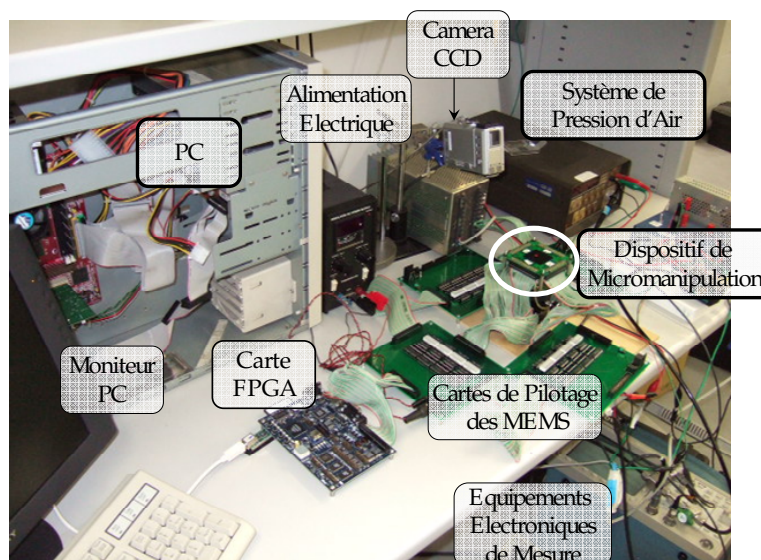


Fig. 6.21 – Banc de validation expérimentale.

La source du flux d'air est fournie par le gaz comprimé d'azote (N_2) à travers des valves de contrôle. Avec un système d'interrupteur adapté et un générateur de fonction, nous pouvons générer une impulsion périodique de flux d'air à 5 Hz de *Marche/Arrêt* afin de réduire l'insta-

bilité aux écoulements d'air sur la surface. Les détails du système de flux à air comprimé sont montrés dans le Chapitre 2.

6.5.1.4 Expérimentation

Dans cette partie, nous présentons les résultats expérimentaux obtenus avec le système de contrôle décentralisé intégré sur le composant FPGA. Toutes les expériences ont été réalisées avec un objet planaire en plastique (dimension : $5,6 \times 5,6 \times 0,25 \text{ mm}^3$, poids : $11,4 \text{ mg}$).

Les lignes et les colonnes se composent des microactionneurs pneumatiques à deux directions : verticaux ou horizontaux, comme l'illustre la figure 6.22. L'arrangement du dispositif distribué se compose de deux parties latérales symétriques composées de 23 lignes de 8 microactionneurs pneumatiques, et de deux parties centrales opposées de 7 colonnes de 12 microactionneurs pneumatiques. Les lignes et les colonnes sont organisées alternativement de microactionneurs pneumatiques verticaux et horizontaux, ce qui permet un actionnement équivalent à quatre directions.

Nous avons constaté que la stabilité du contrôle de mouvement de l'objet dépend de l'orientation initiale de ce dernier. En effet, l'orientation comme indiqué sur la figure 6.22 facilite grandement notre manipulation. En utilisant des images de la caméra CCD, nous pouvons détecter le bord de l'objet, et appliquer par conséquent une poussée adaptée sur celui-ci.

Avec ces éléments de manipulation, nous déciderons de créer une zone de validation où les conditions expérimentales se rapprochent au plus près des descriptions de modélisation, c'est ce que montre la figure 6.22.

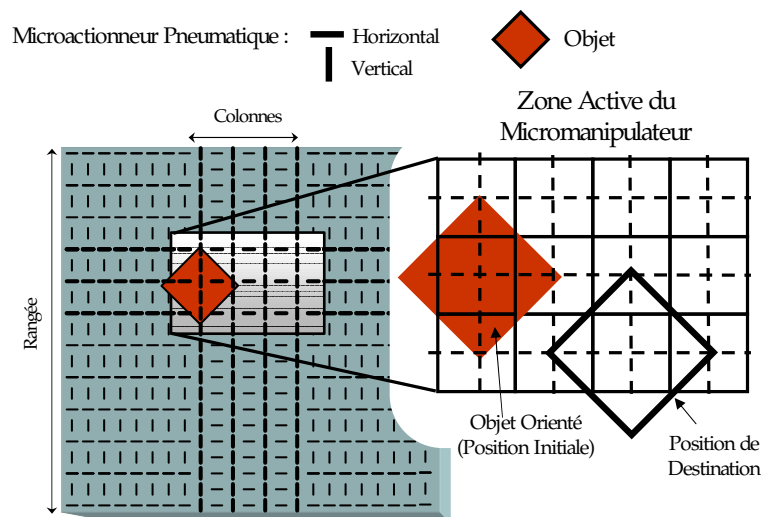


Fig. 6.22 – Configuration expérimentale d'un test de convoyage contrôlé

La figure 6.23 montre des séquences photos du convoyage, extraites d'une vidéo filmée pendant la manipulation expérimentale que nous avons effectuée. La figure 6.23(a) représente l'objet dans sa position initiale et sa position de destination.

Dans la première séquence de convoyage correspondante aux figures 6.23(a) et 6.23(b), nous validons les règles 1, 2, 3 de la stratégie décentralisée de prise de décision. En effet, toutes les cellules couvertes par l'objet ont été activées, et ont toutes pris la même direction pour déplacer l'objet au plus près de la destination. Ainsi, l'objet est bien déplacé horizontalement vers la droite par la rangée de microactionneurs à micro-valve verticales.

Lors de la séquence de convoyage des figures 6.23(b) et 6.23(c), l'objet a le choix entre deux directions pour s'approcher de sa destination. Il choisit la direction sélectionnée par la cellule précédemment activée, respectant la règle 4 de la stratégie décentralisée.

Finalement, dans la dernière séquence de convoyage des figures 6.23(c) et 6.23(d), l'objet est orienté vers sa position de destination par une force verticale. Cette dernière action valide les règles 1, 2, 3, et 6 de la stratégie décentralisée. En effet, une fois que l'objet est arrivé à la position désirée, toutes les cellules activées doivent être désactivées immédiatement, c'est ce que nous avons observé.

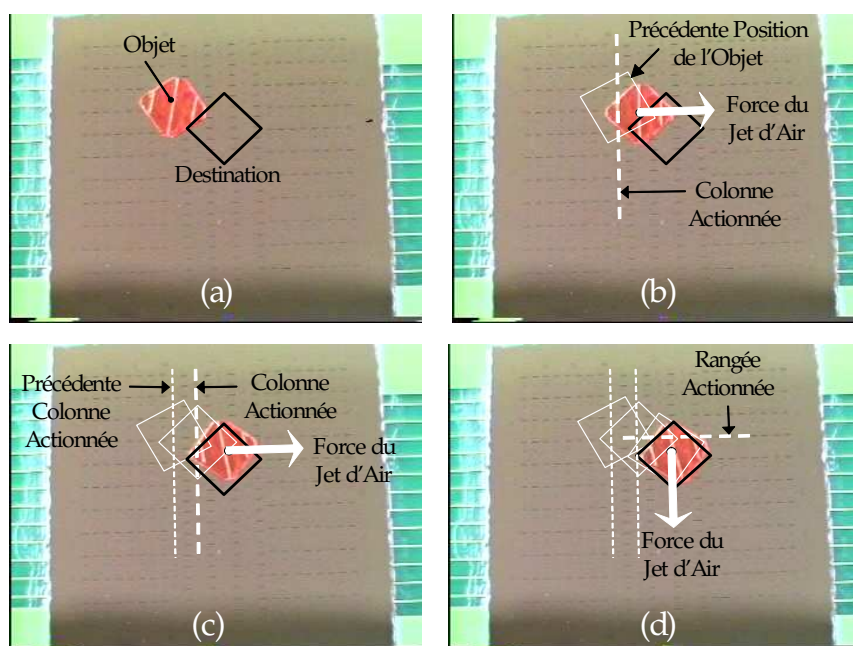


Fig. 6.23 – Résultats expérimentaux de convoyage contrôlé.

L'ensemble de ces résultats d'intégration et expérimentaux de conception, d'implantation et d'exploitation du prototype contrôleur FPGA sont détaillés dans un travail précédent [102].

Si les résultats expérimentaux semblent éloignés des simulations obtenues au niveau « algorithmique » de la méthodologie de validation, c'est que l'étude du paramétrage des conditions expérimentales doit être encore affinée. En effet, les phénomènes expérimentaux observés dans le fonctionnement du micromanipulateur peuvent facilement être rendus dans le modèle VHDL-AMS. Une étude supplémentaire sera donc nécessaire pour cet ajustement.

6.6 Conclusion

Une validation de conception VHDL d'un contrôleur décentralisé entièrement réalisé à partir des modèles VHDL-AMS basée sur les principes du cycle de conception de « Prototypage Virtuel Fonctionnel ». Cette étude a permis de valider au niveau « algorithme », et au niveau « transfert de registre » la fonctionnalité du contrôleur et par la même occasion de mettre en évidence la capacité du modèle VHDL-AMS à gérer plusieurs actions simultanément, fonctionnant de manière distribuée.

A noter que l'on envisage de valider la méthodologie de conception basée sur les principes de « Prototypage Virtuel Fonctionnel », que nous avons présentée ici, pour des modèles multi-langage (C, C++, etc.) qui permettront de co-simuler le logiciel et le matériel.

Conclusions et perspectives

Conclusions

Notre objectif dans cette thèse était de développer et de valider une méthode de modélisation multi-domaines à partir du langage VHDL-AMS pour des structures intelligentes, autonomes et distribuées à base de MEMS. Nous avons appliqué notre étude à un système de micromanipulation, constitué d'une matrice de microactionneurs pneumatiques, élaboré au sein du laboratoire du Pr. Fujita (Fujita Lab.) de l'Université de Tokyo.

Pour mener à bien ces recherches nous avons organisé notre travail en trois phases de développement correspondant aux objectifs de thèse :

1. Établir la première étape de modélisation suivant le cycle de « Prototypage Virtuel Fonctionnel », c'est à dire élaborer le « modèle comportemental » du micromanipulateur distribué à partir du langage VHDL-AMS.
2. Poursuivre l'approche de modélisation basée sur le cycle de « Prototypage Virtuel Fonctionnel » et achever la modélisation multi-domaines du micromanipulateur distribué intégrant les composants « microactionneurs pneumatiques », c'est à dire développer en VHDL-AMS le « modèle structurel comportemental » et les « modèles composants » du dispositif.
3. Élaborer la conception d'un contrôleur décentralisé de commande pour micromanipulateur distribué. Tester l'algorithme élaboré à partir des modèles VHDL-AMS dans le cas du modèle de contrôleur décentralisé en élaborant un flot de conception du « niveau algorithme » jusqu'au « niveau physique ». Puis enfin valider avec succès cette approche sur la plateforme expérimentale installée au sein du Fujita Lab. au Japon.

Bilan des travaux

- Le premier objectif de commande que nous nous étions fixé au début de cette étude a pu être atteint grâce à une stratégie de contrôle décentralisé basée sur des principes de prise de décision simple et adaptée à notre application de micromanipulation distribuée. D'excellents résultats d'intégration ont pu être obtenus laissant présager la possibilité d'une conception à plus grande échelle de ce type de contrôle.

- Nous avons établi avec succès le « modèle comportemental » de la matrice de microactionneurs pneumatiques correspondant au modèle décrit au plus haut niveau d'abstraction de notre hiérarchie de modélisation. Notre second objectif a donc été atteint. Ainsi, après avoir étudié les éléments essentiels du modèle de l'environnement fluide de notre dispositif, nous nous sommes appliqués à suivre une arborescence très précise. Cette approche de description nous permettra de construire notre modèle en utilisant une structure de « briques » réutilisables, ce qui s'adaptera très facilement avec l'établissement du code VHDL-AMS lors de l'élaboration du programme du modèle. Nous noterons à cet effet, la grande flexibilité de ce langage. Les résultats de simulation du « modèle comportemental » de la matrice viendront corroborer les attentes fonctionnelles. De manière plus spectaculaire, nous réussirons à faire coïncider par simulation la trajectoire, obtenue expérimentalement en boucle ouverte, d'un objet convoyé en lévitation sur la surface active de la matrice.

A l'inverse, certains objectifs de développement du « modèle comportemental » n'ont pas été atteints par manque de temps. C'est cas d'étude comme « la modélisation 3D avec des rotations dans les trois plans », « le cas d'objets de forme quelconque, de taille quelconque, etc. ». Ces travaux qui ne présentent pas de difficultés conceptuelles manquent pourtant à l'achèvement du « modèle comportemental ». Ils doivent encore être développés.

- Le « modèle structurel comportemental » a pu être développé et validé au niveau « modèle structurel composant » en intégrant le composant décrivant le modèle du microactionneur pneumatique. Les résultats de simulation obtenus sous le logiciel SMASH ont répondu à nos attentes tant sur le plan « modèle » que sur celui des performances du micromanipulateur. Le flot complet de modélisation semble bien maîtrisé et ces recherches s'avèrent très prometteuses pour valider notre approche dans la méthodologie de « Prototypage Virtuel Fonctionnel » développée par *Yannick Hervé* au sein de l'InESS et valorisée par la société Systems'VIP.
- L'approche de modélisation VHDL-AMS basée sur les principes de « Prototypage Virtuel Fonctionnel » a été testée en élaborant un flot de conception du « niveau algorithmique » jusqu'au « niveau physique ». Le modèle de contrôleur décentralisé intégré sur cible FPGA a été ainsi validé à tous ses niveaux de description avant son implémentation.

Apports originaux

- Les développements concernant l'objectif de contrôle ont marqué une première, à notre connaissance, par sa démarche et ses aboutissements. Ces travaux ont été publiés dans plusieurs revues et conférences internationales. Parmi lesquelles nous citerons la revue internationale « *IEEE Transactions on Industrial Electronics* » où nous avons été sélectionnés parmi plus de 50 soumissions du domaine dans le « *Special section on FPGAs used in industrial control systems* » d'août 2007.
- Appliquer un flot de modélisation VHDL-AMS, basé sur les principes du cycle de concep-

tion de « Prototypage Virtuel Fonctionnel », pour la simulation de structures intelligentes, autonomes et distribuées à base de MEMS était un défi technologique. Il a été relevé grâce à la convergence de plusieurs facteurs : une méthodologie très performante, un langage VHDL-AMS efficace et flexible à un haut niveau de manipulation, des outils associés performants (SMASH, version 7.0.) et une méthode perfectionnée d'appréciation des modèles. En tous points, ces développements se révèlent originaux et innovants pour les développements à base de MEMS. Ces résultats ont été publiés dans deux conférences internationales et ont été soumis dans la revue « Sensors and Actuators A : Physical (impact factor - 2006 JCR Science Edition : 1,434) ».

- Une validation de conception VHDL entièrement réalisée à partir des modèles VHDL-AMS basé sur les principes du cycle de conception de « Prototypage Virtuel Fonctionnel ».

Perspectives

- La puissance du langage VHDL-AMS peut-être exploitée de façon encore plus efficace que dans les premiers travaux de modélisation dont nous rendons compte dans cette thèse. Il sera donc essentiel de réaliser les dernières tâches de modélisation pour valoriser toute les possibilités du langage VHDL-AMS dans le cas de tels développements. Par exemple, nous envisageons de d'améliorer les techniques de configuration (CONFIGURATION) et de généricité (GENERATE) du « modèle structurel comportemental ». Nous travaillerons aussi sur la modélisation 3-D avec des rotations de l'objet dans les trois plans mais aussi le cas d'objets de forme quelconque, de taille quelconque, etc.
- La méthodologie de conception « Prototypage Virtuel Fonctionnel » devra être validée pour des modèles multi-langages (C, C++, etc.) qui permettront de co-simuler le logiciel et le matériel.

Il s'agira par exemple d'interfacer des modèles de haut niveau pour vérifier notre contrôleur VHDL à tous les niveaux de conception :

1. Modèle algorithmique (éventuellement en C)
2. Modèle RTL VHDL (ou system-C) synthétisable
3. Modèle synthétisé (et éventuellement rétro annoté)

A chaque niveau de conception, nous appliquerons le modèle de co-simulation approprié basé sur notre flot de modélisation VHDL-AMS et « Prototypage Virtuel Fonctionnel » pour valider la réponse aux spécifications du composant.

En cours de thèse, nous avons été sollicités par nos collègues du LAB¹ et du LIFC² de Besançon pour élaborer un projet de recherche et répondre à l'appel du programme « Systèmes Interactifs et Robotique » de l'ANR³ (nous rajouterons également la contribution du LAAS pour

¹Laboratoire d'Automatique de Besançon

²Laboratoire d'Informatique de l'Université de Franche-Comté

³Agence Nationale de la Recherche

les aspects « décentralisés »). Notre contribution dans le domaine des modèles multi-physiques et multi-technologies à travers cette thèse a été perçue comme favorable à la réussite de notre intégration dans le projet. Celui-ci a été sélectionné en octobre 2006 et fait l'objet d'un financement depuis avril 2007 par l'ANR. Ces recherches s'articuleront autour du projet intitulé « Smart Surface » dont le financement s'étend de 2007 à 2010 et prévoit à termes le développement de nouveaux prototypes de micromanipulateurs 2-D à base de microactionneurs pneumatiques en technologie MEMS.

Une position de chercheur post-doctorant de seize mois a été attribuée à notre équipe de recherche de l'InESS dans le domaine de la modélisation multi-domaines. L'objectif principal de ces recherches concerne l'exploitation du flot de modélisation VHDL-AMS, basé sur les principes du cycle de conception de « Prototypage Virtuel Fonctionnel » pour valider des tâches de contrôle avancées telles que :

1. Adaptive and distributed control using reinforcement learning ;
2. Distributed information management.

J'ai l'honneur d'avoir été choisi sur ce poste de chercheur post-doctorant de seize mois. J'aurai ainsi l'occasion de poursuivre mes travaux de thèses à travers un projet de recherche de très haut niveau.

Références bibliographiques

- [1] A. Berlin and K. Gabriel, “Distributed MEMS : New challenges for computation,” *IEEE Computational Science and Engineering Journal*, vol. 4(1), pp. pp. 12–16, March 1997.
- [2] K. F. Böhringer, B. R. Donald, L. E. Kavraki, and F. Lamiraux, “A distributed, universal device for planar parts feeding : unique part orientation in programmable force fields,” in *Distributed Manipulation*, K.-F. Böhringer and H. Choset, Eds. Kluwer Academic Publishers, 2000.
- [3] M. H. Yim, J. E. Reich, and A. A. Berlin, “Two approaches to distributed manipulation,” in *Distributed Manipulation*, E. Choset and K. Bohringer, Eds. Kluwer Academic Publishers, 2000.
- [4] M. P. J. Fromherz and W. B. Jackson, “Force allocation in a large-scale distributed active surface,” *IEEE Transactions on Control Systems Technology*, vol. 1(5), pp. pp. 641–655, 2003.
- [5] L. Demailly, “Contrôle distribué en optique adaptative. Vers un système multi-agents intelligent,” Ph.D. dissertation, Université de Caen, 1996.
- [6] F. Pecheux, C. Lallement, and A. Vachoux, “VHDL-AMS and Verilog-AMS as alternative hardware description languages for efficient modeling of multidiscipline systems,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 24, no. 2, pp. 204–225, Feb. 2005.
- [7] J. Bryzek, S. Roundy, B. Bircumshaw, C. Chung, K. Castellino, J. R. Stetter, and M. Vestel, “Marvelous MEMS - Advanced IC sensors and microstructures for high volume applications,” *IEEE Circuits & Devices Magazine*, vol. March/April, pp. 8–28, 2006.
- [8] “MEMS clearinghouse, what is MEMS technology? : <http://www.memsnet.org/mems/what-is.html>.”
- [9] M. ed Hak, Ed., *The MEMS Handbook*. CRC Press, 2002.
- [10] “<http://www.electronics.ca/presscenter/articles/>, Electronics CA Publications, MEMS Market To Reach \$ 12.5 Billion In 2010.”
- [11] D. S. Eddy and D. R. Sparks, “Application of MEMS technology in automotive sensors and actuator,” in *Proceedings of IEEE*, vol. Vol. 86, 1998, pp. pp. 1747–1755.

- [12] P. F. V. Kessel, L. Hornbeck, R. Meier, and M. Douglas, "A MEMS based projection display," in *Proceedings of IEEE*, vol. Vol. 86, 1998, pp. pp. 1687–1704.
- [13] L. Smith, A. Soderbarg, and U. Bjorkengren, "Continuous ink-jet print head utilizing silicon micromachined nozzles," *Sensors and Actuators A*, vol. 43, pp. pp. 256–261, 1994.
- [14] H. C. Kim and K. Chun, "RF MEMS technology," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 2(3), pp. pp. 249–261, May 2007.
- [15] C. Bergaud, "Micro et nanosystèmes mécaniques : problématiques afférentes à la réduction en taille et aux interfaces pour des applications dans le domaine de la chimie et de la biologie," Ph.D. dissertation, Institut National des Sciences Appliquées de Toulouse, 2005.
- [16] T. Adrega, D. Prazeres, V. Chu, and J. Conde, "Thin-film silicon MEMS DNA sensors," *Journal of Non-Crystalline Solids*, Vol.352, vol. Vol.352, Issues 9-20, pp. pp. 1999–2003, 2006.
- [17] R.B. Fair, et al., "Special issues on biomedical applications for MEMS and microfluidics," in *Proceedings of IEEE*, vol. 92(1), 2004, pp. pp. 3–5,.
- [18] R. Langer et al., "Review : MEMS technology for physiologically integrated devices," in *Proceedings of IEEE*, vol. 92(1), 2004, pp. pp. 6–21,.
- [19] J.C. Eloy - CEO - Yole Développement, "Global MEMS market evolution," in *The 14th International Conference of Solid-State Sensors, Actuators and Microsystems (TRANSDUCER'07)*, 2007.
- [20] P. Minotti, G. B. P. L. Moal, and E. Joseph, "Réseaux de micro-actionneurs sur silicium et micromanipulateurs parallèles," in *Micro-actionneurs électroactifs*, O. Cugat, Ed. Paris, France : Hermès Science, 2002, vol. Chapitre 4.
- [21] S. Mir and S. Martinez, "Introduction aux microsystèmes sur silicium," in *Conception des microsystèmes sur silicium*, S. Mir, Ed. Paris, France : Hermès Science, 2002, ch. Chapitre 1.
- [22] F. C. P. Jänker, "New actuators for aircraft and space applications," in *in proceeding of The 10th International Conference on New Actuators (ACTUATOR 2006)*, 2006.
- [23] M. Krüger, C. U. Grosse, and P. J. Marrón, "Wireless structural health monitoring using MEMS," in *Damage Assessment of Structures*, W. M. O. et al, Ed. Proc. Intern. Symp., Gdansk, Poland, Zürich : Trans Tech., 2005.
- [24] J. S. M. D. Wang and S. Nie, "Biomedical nanotechnology with bioinformatics - the promise and current progress," in *Proceedings of IEEE*, vol. 95(7), 2007, pp. pp. 1386–1389.
- [25] S. Chowdhury, G. Jullien, M. Ahmadi, and W. Miller, "A MEMS micromagnetic actuator for use in a bionic interface," in *The 2000 IEEE International Symposium on Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva.*, vol. 5, 28-31 May 2000, pp. 181–184.
- [26] M. Allen, M. Raulli, K. Maute, and D. M. Frangopol, "Reliability-based analysis and design optimization of electrostatically actuated MEMS," *Computers & Structures*, vol. Vol. 82, Issues 13-14, pp. 1007–1020, May 2004.

- [27] “Merriam websters dictionary : <http://www.merriam-webster.com>.”
- [28] R. A. Breckenbridge and C. Husson, “Smart sensors in spacecraft : The impact and trends,” in *Proceeding of The IAA/NASA Conference on Smart Sensors*, 1978, pp. pp. 1–5.
- [29] J. W. Gardner, V. K. Varadan, and O. Awadelkarim, *Microsensors MEMS and smart devices*. England : John Wiley & Sons, LTD, 2001.
- [30] B. W. Cook, S. Lanzisera, and K. S. J. Pister, “SoC issues for RF smart dust,” in *Proceeding of the IEEE*, vol. 94(6), June 2006, pp. pp. 1177–1195.
- [31] J.-P. Jamont, “Diamond : Une approche pour la conception de systèmes multi-agents embarqués,” Ph.D. dissertation, Institut National Polytechnique de Grenoble, 2005.
- [32] J. Ferber, *Les systèmes multi-agents, Vers une intelligence collective*, Dunod, Ed. Inter-Editions, Septembre 1995.
- [33] O. C. Zienkiewicz, R. L. Taylor, and J. Zhu, *The Finite Element Method : Its Basis and Fundamentals*. Butterworth-Heinemann ; 6 edition, March 2005.
- [34] S. Muratet, “Conception, caractérisation et modélisation : Fiabilité prédictive de MEMS à actionnement électrothermique,” Ph.D. dissertation, Institut National des Sciences Appliquées de Toulouse, 2005.
- [35] J. Lienemann, B. R. B. Evgenii, J. G. Korvink, and Ferber, “MST MEMS compact modeling meets model order reduction : Requirements and benchmarks,” « *Special Issue on Order Reduction of Large-Scale Systems* », *Linear Algebra and its Applications*, vol. Vol. 415, Issues 2-3, pp. 469–498, June 2006.
- [36] P. Desgreys, “Composant optoélectronique : diode laser VCSEL (vertical cavity surface emitting laser),” in *Modélisation de systèmes complexes avec VHDL-AMS*, EPFL, Ed. Lausanne, Suisse : Presses Polytechniques Universitaires Romandes, 2004.
- [37] Y. Fukuta, Y.-A. Chapuis, Y. Mita, and H. Fujita, “Design, fabrication, and control of MEMS-based actuator arrays for air-flow distributed micromanipulation,” *Journal of Microelectromechanical Systems*, August 2006.
- [38] K.-F. Böhringer and H. Choset, *Distributed Manipulation*. Kluwer Academic Publishing, 2000.
- [39] J. Luntz and W. Messner, “Dynamics and control of discrete distributed manipulation,” in *Proceedings of the 2001 IEEE International Conference on Control Applications (CCA '01)*, 5-7 Sept. 2001, pp. 1111–1116.
- [40] K. Pister, R. Fearing, and R. Howe, “A planar air levitated electrostatic actuator system,” in *Micro Electro Mechanical Systems, 1990. Proceedings, An Investigation of Micro Structures, Sensors, Actuators, Machines and Robots. , IEEE*, 11-14 Feb. 1990, pp. 67–71.
- [41] H. Fujita, “Group work of microactuators,” in *Proc. International Advanced Robot Program Workshop on Micromachine Technologies and Systems*, Tokyo, Japan, 1993, pp. 24–31.

- [42] K.-F. Bohringer, B. Donald, and N. MacDonald, "Single-crystal silicon actuator arrays for micro manipulation tasks," in *Micro Electro Mechanical Systems, 1996, MEMS 96, Proceedings. An Investigation of Micro Structures, Sensors, Actuators, Machines and Systems. IEEE, The Ninth Annual International Workshop on*, 11-15 Feb. 1996, pp. 7–12.
- [43] M. Ataka, A. Omodaka, N. Takeshima, and H. Fujita, "Fabrication and operation of polyimide bimorph actuators for a ciliary motion system," *Microelectromechanical Systems, Journal of*, vol. 2, no. 4, pp. 146–150, Dec. 1993.
- [44] S. Konishi and H. Fujita, "A conveyance system using air flow based on the concept of distributed micro motion systems," *Microelectromechanical Systems, Journal of*, vol. 3, no. 2, pp. 54–58, June 1994.
- [45] H. Nakazawa, Y. Watanabe, O. Morita, M. Edo, and E. Yonezawa, "The two-dimensional micro conveyer : principles and fabrication process of the actuator," in *Proc. of The International Conference on Solid State Sensors and Actuators (TRANSDUCERS '97)*, vol. Vol. 1, Chicago, U.S, June 1997, pp. pp. 33–36.
- [46] T. Ebefors and G. Stemme, "Microrobotics," in *The MEMS Handbook*, M. G. el Hak, Ed. CRC Press LLC, 2002, ch. Chapter 28.
- [47] P. Minotti, G. Bourbon, P. Langlet, and T. Masuzawa, "Arrayed electrostatic scratch drive actuators : Towrads visible effects up to the human scale," *Journal of Intelligent Material Systems and Structures*, vol. 9(10), pp. pp. 837–846, 1998.
- [48] T. Ebefors and G. Stemme, "A robust micro conveyer realized by arrayed polyimide joints actuators," *Journal of Micromechanical and Microengineering*, vol. 10(3), pp. pp. 337–349, 2000.
- [49] J. W. Suh, R. B. Darling, K.-F. Bohringer, B. R. Donald, H. Baltes, and G. T. A. Kovacs, "CMOS integrated ciliary actuator array as a general-purpose micromanipulation tool for smallobjects," *IEEE/ASME Journal of Microelectromechanical Systems*, vol. 8(4), pp. pp. 483–496, 1999.
- [50] Y.-A. Chapuis, E. Sarajlic, and H. Fujita, "Hybrid polymer/silicon technology for thermal pneumatic microactuator used in distributed air-jet planar micromanipulation," in *Proc. of The 21st IEEE International Conference on Micro Electro Mechanical Systems*, Tucson, Arizona, USA, January 13-17 2008 (à paraître).
- [51] C. Chang, C.-F. Chiang, and C.-H. Liu, "A lobster-sniffing-inspired method for micro-objects manipulation using electrstatic micro-actuators," *Journal of Micromechanical and Microengineering*, vol. Vol. 54, pp. pp. 812–821, 2005.
- [52] T. Strick, J.-F. Allemand, V. Croquette, and D. Bensimon, "The manipulation of single biomolecules," *Physical Today*, vol. Vol. 15, pp. pp. 46–60, 2001.
- [53] K.-F. Bohringer, "Modeling and controlling parallel tasks in droplet-based microfluidic systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25(2), pp. pp. 334– 344, 2006.

- [54] S. K. Cho, H. Moon, and C.-J. Kim, "Creating, transporting, cutting, and merging liquid droplets by electrowetting-based actuation for digital microfluidic circuits," *Journal of Microelectromechanical Systems*, vol. 12(1), pp. pp. 70–80, 2003.
- [55] L. Zhou, Y. Herve, Y.-A. Chapuis, and H. Fujita, "VHDL-AMS modeling for simulation of MEMS array-based smart surface applied in microfluid air-flow environment," in *Sensors, 2006. 5th IEEE Conference on*, Oct. 2007, pp. 807–810.
- [56] T. Riesgo, Y. Torroja, and E. de la Torre, "Design methodologies based on hardware description languages," *IEEE Transactions on Industrial Electronics*, vol. 46, pp. 3 – 12, 1999.
- [57] P. Schneider, A. Scheider, and P. Schwarz, "A modular approach for simulation-based optimization of MEMS," *Microelectronic journal*, vol. 33, pp. 29–38, 2002.
- [58] P. V. Nikitin and C.-J. R. Shi, "VHDL-AMS based modeling and simulation of mixed-technology microsystems : a tutorial," *Integration, the VLSI journal*, vol. 40, pp. 261–273, 2007.
- [59] Y. Hervé, "Functional Virtual Prototyping – Design Flow and VHDL-AMS," in *Proc. of Forum on Specification of Design Languages (FDL'06)*, 2006.
- [60] G. K. Fedder, "Simulation of microelectromechanical systems," Ph.D. dissertation, Faculty of Electrical Engineering and computer sciences, University of California, Berkeley CA, 1994.
- [61] Y. Hervé, *VHDL-AMS : Applications et enjeux industriels*. DUNOD, 2002.
- [62] J. Weber and M. Meaudre, *Le langage VHDL*. DUNOD, 2001.
- [63] Y. Hervé, "VHDL-AMS : Un atout pour la conception de système," *IEEE*, 2002.
- [64] A. Daboli and R. Vemuri, "Behavioral modeling for high-level synthesis of analog and mixed-signal systems from VHDL-AMS," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 22, no. 11, pp. 1504–1520, Nov. 2003.
- [65] P. Nikitin, C. Shi, and B. Wan, "Modeling partial differential equations in VHDL-AMS [mixed signal systems applications]," in *SOC Conference, 2003. Proceedings. IEEE International [Systems-on-Chip]*, 17-20 Sept. 2003, pp. 345–348.
- [66] A. Vachoux, *Modélisation de systèmes analogiques et mixtes - Introduction à VHDL-AMS*, Ecole Polytechnique Fédérale de Lausanne, Eté 2003, notes de cours à option 2ème cycle - Version 2003. [Online]. Available : <http://lsmwww.epfl.ch/Education/former/2002-2003/modelmix03/Documents/modelmixte02.book.pdf>
- [67] "<http://www.eda.org/verilog-ams/>," Novembre 2004.
- [68] "<http://www.synopsys.com/products/mixedsignal/saber/saber.html>," 2007.
- [69] "<http://www.modelica.org>," 2007.
- [70] "<http://www.mathworks.fr/products/matlab/>," 2007.

- [71] S. Guessab and J. Oudinot, "How can MATLAB/Simulink co-exist with VHDL-AMS?" in *Euro DesignCon*, 2004.
- [72] D. GUIHAL, "Modélisation en langage VHDL-AMS des systèmes pluridisciplinaires," Ph.D. dissertation, Université Toulouse III, 2007.
- [73] P. Wilson, J. Ross, A. Brrown, and A. Rushton, "Multiple domain behavioral modeling using VHDL-AMS," *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium*, vol. 5, pp. V-644V-647, May 2004.
- [74] R. Guelaz, D. Kourtiche, M. Nadi, and Y. Herve, "Ultrasonic piezoceramic transducer modeling with VHDL-AMS IEEE 1076.1," in *Sensors, 2004. Proceedings of IEEE*, 24-27 Oct. 2004, pp. 87-90vol.1.
- [75] M. Sida, R. Ahola, and D. Wallner, "Bluetooth transceiver design and simulation with VHDL-AMS," *The Chip*, March 2003.
- [76] 2007. [Online]. Available : <http://www.systemsvip.com>
- [77] E. Christen and K. Bakalar, "VHDL-AMS - a hardware description language for analog and mixed-signal applications," *Circuits and Systems II : Analog and Digital Signal Processing, IEEE Transactions on [see also Circuits and Systems II : Express Briefs, IEEE Transactions on]*, vol. 46, no. 10, pp. 1263-1272, Oct. 1999.
- [78] "<http://www.ansoft.com/products/em/simplorer/>," 2007.
- [79] "http://www.mentor.com/products/ic_nanometer_design/ms_circuit_simulation/advance_ms/index.cfm," 2007.
- [80] M. Schlegel, F. Bennini, J. Mehner, G. Herrmann, D. Muller, and W. Dotzel, "Analyzing and simulation of MEMS in VHDL-AMS based on reduced-order FE models," *Sensors Journal, IEEE*, vol. 5, no. 5, pp. 1019-1026, Oct. 2005.
- [81] "<http://www.dolphin-integration.com/>," september 2007.
- [82] "Synopsys®. « `saber hdl` : Language-independant mixed-signal multi-technology simulator ». synopsys® 2003." [Online]. Available : http://www.synopsys.com/products/mixedsignal/saber/saber_hdl_ds.html
- [83] P. Chassaing, *Mécanique des fluides - Eléments d'un premier parcours*. Cépaduès-Editions (Collection Polytech), 2000.
- [84] R. Comolet, *Mécanique des fluides expérimentale, Tome II, Dynamique des Fluides réels*. Editions Masson, 1994.
- [85] Y. A. Cencel and J. M. Cimbala, "Flow over bodies : drag and lift," in *Fluid Mechanics - Fundamentals and Applications*. McGraw - Hill International Edition, 2006, ch. 11.
- [86] O. Ducloux, "Microsystèmes magnéto-mécaniques (MMMS) pour le contrôle actif d'écoulements aérauliques," Ph.D. dissertation, Ecole centrale de Lille and Université de Valenciennes et du Hainaut Cambresis, 2006.

- [87] Z. Toffano, M. Pez, P. Desgreys, Y. Herve, C. Le Brun, J.-C. Mollier, G. Barbary, J.-J. Charlot, S. Constant, A. Destrez, M. Karray, M. Marec, A. Rissons, and S. Snaidero, "Multilevel behavioral simulation of VCSEL-based optoelectronic modules," *Selected Topics in Quantum Electronics, IEEE Journal of*, vol. 9, no. 3, pp. 949–960, May-June 2003.
- [88] R. Shina, C. J. J. Paredis, and P. K. Khosla, "Interaction modeling in systems design," in *Proc. of ASME 2001 Design Engineering Technical Conference and Computers and Information in Engineering Conference (DETC'01)*, 2001, pp. 1–9.
- [89] B. Chen and J. Miao, "Influence of deep RIE tolerances on comb-drive actuator," *Journal of Physics D : Applied Physics*, vol. 40, pp. 970–976, 2007.
- [90] B. Borovic, F. L. Lewis, A. Q. Liu, E. S. Kolesar, and Popa, "The lateral instability problem in electrostatic comb drive actuators : modelling and feedback control," *Journal of Micromechanics and Microengineering*, vol. 16, pp. 1233–1241, 2006.
- [91] J. J. Rodriguez-Andina, M. J. Moure, and M. D. Valdes, "Features, design tools, and application domain of FPGAs," *IEEE Transactions on Industrial Electronics, Special Section on FPGAs Used in Industrial Control Systems*, vol. 54(4), pp. 1810 – 1823, 2007.
- [92] A. Jerraya, H. Ding, P. Kission, and M. Rahmouni, *Behavioral Synthesis and Component Reuse with VHDL*. Kluwer Academic Publishers, 1998.
- [93] E. Monmasson and Y.-A. Chapuis, "Apports des FPGA dans la commande des systèmes électriques," in *Commande des Systèmes Electriques - Perspectives Technologiques*, L. Loron, Ed. Hermès, 2003.
- [94] —, "Contributions of FPGAs to the control of electrical systems," *IEEE Industrial Electronics Society Newsletter*, vol. 49, pp. 8–15, 2002.
- [95] J. David, "Efficient functional verification for mixed signal ip," in *Behavioral Modeling and Simulation Conference, 2004. BMAS 2004. Proceedings of the 2004 IEEE International*, 21-22 Oct. 2004, pp. 53–58.
- [96] A. Danesi, A. Fagiolini, I. Savino, L. Pallottino, R. Schiavi, G. Dini, and A. Bicchi, "A scalable platform for safe and secure decentralized traffic management of multiagent mobile systems," in *Proc. of the ACM Workshop on Real-World Wireless Sensor Networks*, 2006.
- [97] F. Zhao, "Wireless sensor networks : A new computing platform for tomorrow's internet," in *Proc. of the IEEE 6th CAS Symposium on Emerging Technologies : Mobile and Wireless Communication*, Shanghai, China, May 31-June 2 2004.
- [98] S. Wolfram, *Cellular Automata and Complexity*. Addison Wesley. Reading, 1994.
- [99] J. T. Feddema, C. Lewis, and D. A. Schoenwald, "Decentralized control of cooperative robotic vehicles : Theory and application," *IEEE Transactions on Robotics and Automation*, vol. 18, 2002.
- [100] H. Takahashi and K. Ohnishi, "Autonomous decentralized control for formation of multiple mobile-robots considering ability of robot," in *Industrial Electronics Society, 2003. IECON '03. The 29th Annual Conference of the IEEE*, vol. 3, 2-6 Nov. 2003, pp. 2041–2046Vol.3.

- [101] L. Zhou, Y.-A. Chapuis, J.-P. Blondé, H. Berviller, Y. Fukuta, and H. Fujita, “Integrated control strategy for autonomous decentralized conveyance systems based on distributed MEMS arrays,” in *Proc. SPIE 5383, 2004, SPIE 11th Annual International Symposium on Smart Structures and Materials*, San Diego (USA), March 14-18 2004.
- [102] Y.-A. Chapuis, L. Zhou, Y. Fukuta, Y. Mita, and H. Fujita, “FPGA-based decentralized control of arrayed mems for microrobotic application,” *IEEE Transactions on Industrial Electronics, Special Section on FPGAs Used in Industrial Control Systems*, vol. 54(4), pp. 1926 – 1936, 2007.

Liste de publications

Publications dans des revues internationales à comités de lecture :

- Y.-A. Chapuis, **L. Zhou**, Y. Fukuta, Y. Mita, and H. Fujita, “FPGA-based decentralized control of arrayed mems for microrobotic application,” *IEEE Transactions on Industrial Electronics, Special Section on FPGAs Used in Industrial Control Systems*, vol. 54(4), pp. 1926 – 1936, 2007.
- Y.-A. Chapuis, **L. Zhou**, Y. Fujita, Y. Hervé, “Multi-domain simulation using VHDL-AMS for distributed MEMS in air-fluid environment : Case of a 2-D air-jet micromanipulator,” *submitted at Sensors and Actuator A*, 2007.

Publications dans des conférences internationales à comités de lecture

- **L. Zhou**, Y. Herve, Y.-A. Chapuis, and H. Fujita, “VHDL-AMS modeling for simulation of MEMS array-based smart surface applied in microfluid air-flow environment,” in *Sensors, 2006. 5th IEEE Conference on*, Oct. 2007, pp. 807–810.
- **L. Zhou**, Y.-A. Chapuis, and Y. Hervé, “VHDL-AMS modeling and simulation of MEMS-based electrostatic actuator for microvalve,” in *The Forum on specification & Design Languages (FDL’06)*, 2006, pp. 41–45.
- **L. Zhou**, Y.-A. Chapuis, Y. Fukuta, Y. Mita, F. Braun, and H. Fujita, “Architecture and implementation of distributed control system for MEMS-based intelligent motion surface,” in *IEEE International Symposium on Industrial Electronics*, Dubrovnik (Croatia), June 20-23 2005, pp. 1043–1048.
- **L. Zhou**, Y.-A. Chapuis, J.-P. Blondé, H. Berviller, Y. Fukuta, and H. Fujita, “Integrated control strategy for autonomous decentralized conveyance systems based on distributed MEMS arrays,” in *Proc. SPIE 5383, 2004, SPIE 11th Annual International Symposium*

on *Smart Structures and Materials*, San Diego (USA), March 14-18 2004.

Publications dans des conférences nationales à comités de lecture

- **L. Zhou**, Y.-A. Chapuis, and Y. Hervé, “Simulation multi-physique et multi-technologique de microstructures type smart MEMS en milieu réel à partir du langage VHDL-AMS,” in *Journées Nationales du Réseau Doctoral en Microélectronique*, Lille, 2007.
- **L. Zhou**, Y.-A. Chapuis, and Y. Hervé, “Modélisation VHDL-AMS haut niveau de micro-systèmes distribués pour surface microfluidique activée,” in *Journées Nationales du Réseau Doctoral en Microélectronique*, Rennes, 2006.
- **L. Zhou** and Y.-A. Chapuis, “Conception de micro-structures intelligentes à base de micro-systèmes décentralisées et autonomes combinant technologies MEMS et microélectronique,” in *Journées Nationales du Réseau Doctoral en Microélectronique*, Paris, 2005.
- Y.-A. Chapuis, Y. Fukuta, **L. Zhou**, Y. Mita, and H. Fujita, “Utilisation des technologies MEMS pour la conception de surfaces actives microfluidiques appliquées à la micromanipulation,” in *Journées Scientifiques Francophones*, Tokyo, 2004.

Annexes

Annexe A

Codes source VHDL-AMS : modèle structurel comportemental

```
>>> VHDL

--
--           Micro Fluidic Conveyor HEADER - Do not remove
--*****
-- Description : This model describes the behaviors of a
-- microconveyor transporting an object.
--
-- File :                               Owned by :
--*****
-- Author(s) : Lingfei ZHOU             Organisation : InESS
-- Date :                               Project :
--
-- Current Version : 0.70               Verified by :
-- TRT validation :
-- History :
-- Version:          Date: 25/07/2007   Modification :      By:
--
-- SAMSH 5.7.0 Lingfei ZHOU
--
-- Associated files :
--
-- Remarks :
--
-- This model is developed by basing on the model 0.59.
-- The difference with the model 0.59 : X and Y axis, see the figure.
--
-- This model is structural. With the model of the valve.
--
-- Limitations / Field of use :
--
-- 3D case
--
-- The object's movement is assimilated with its gravity's center;
-- the force of the flow is homogeneous everywhere on the object;
-- do not take care of the rotation of the objetct;
```

```

-- three dimensions: vertical and axis X, Y;
-- the object rebounds when it hits the floor;
-- the object can covers several holes in a same time;
--
-- There is a control strategy in this model.
--
-- Remarks:
-- The form of the object:
--
--      top edge
--=====
--| |
--| |
--| |
--left edge |      Object | right edge
--| |
--| |
--| |
--=====
--      bottom edge
--
--
-- The layout of the cells:
--.
--.
--.
--|||||
-------
--|||||
-------
--..... ||||| .....
-------
--|||||
-------
--|||||
--.
--.
--.
--
-- Bibliographic reference :
--
--*****
--*****
-- My Package
--*****

library ieee;
use ieee.std_logic_1164.all;
package new_type is
type dirflu is array (1 to 40) of std_logic_vector(1 downto 0); -- modif 19/07
type dirflux is array (1 to 40) of dirflu; -- ajout 19/07
type en_op is array (1 to 40) of bit_vector(1 to 40); -- ajout 19/07
end new_type;

```

```

--*****
-- Microactuator
--*****

library ieee;
use ieee.math_real.all;
use ieee.std_logic_1164.all;
use ieee.electrical_systems.all;
use ieee.mechanical_systems.all;
use work.new_type.all;
use work.all;

entity Valve IS
    generic( m: real; -- The movable electrode's mass
kv: real; -- The air's viscosity
rd: real; -- shock output
st: real; -- The distance between the movable electrode
-- and the stopper.
epsilon:real; -- The dielectric constant of vacuum
e_young:real; -- Young's Modulus(silicon);
lv: real; -- The length of vertical suspending beam
h: real; -- The thickness of suspending beams
g0: real; -- The intial gap between electrodes
w: real; -- The suspending beam's width

vi: real -- The supply voltage.
);

port ( enable : in bit; -- From "Microcontroler".
direction: in std_logic_vector(1 downto 0);-- The direction of the flux.
op : out bit -- For "Interface".
);
end Valve;

-----
-- First shape
-----

architecture form1 of Valve is

quantity x: real; -- The motion of the suspending beam
quantity v: real; -- The velocity of the suspending beam
quantity fe: real; -- The electrostatic force
quantity fr: real; -- The resistance.
quantity keq: real; -- The spring stiffness;
quantity vpullin: real; -- The voltage Vpull-in
signal gap1 : boolean;
signal gap2 : boolean;

BEGIN
gap1 <= not x'above(-st);
gap2 <= x'above(st);

BREAK v => -rd*v*0.9 WHEN gap1 or gap2;

```

```

break on gap1;
break on gap2;

x'DOT == v;  -- The suspending beam's velocity.

if enable = '1' use
if direction = "01" use
fe == 0.5*epsilon*lv*h*vi**2/(g0-abs(x))**2;
else
fe == -0.5*epsilon*lv*h*vi**2/(g0-abs(x))**2;
end use;
else
fe == 0.0;
end use;

-- Forme1
0.0625*keq == e_young*h*w**3/(56.0*lv**3);

-- fr == keq*x+kv*v; -- The resistance with the viscosity.
fr == keq * x; -- The resistance without the viscosity.

m * v'dot == fe - fr;

-- Forme 1
vpullin == sqrt(8.0*g0**3*keq/(27.0*epsilon*lv*h));

process
begin
wait on enable;
if enable = '1' then
op <='1';
else
op <='0';
end if;
end process;

END;

--*****
--Microsensor
--*****

library ieee;
use ieee.math_real.all;
use ieee.std_logic_1164.all;
use ieee.electrical_systems.all;
use work.new_type.all;
use work.all;

entity microsensor is

generic ( x,y : integer -- Microsensor's Coordinates. -- modif 19/07
);

port ( n1,n2,n3,n4 : in integer; -- From "Interface"; -- modif 19/07

```

```

arrival : out bit -- For "Microcontroler";
);
end entity microsensors;

architecture structural of microsensors is

begin
process
begin
wait on n1,n2,n3,n4;
if x >= n2 and x <= n1 and y >= n4 and y <= n3 then -- Under the object, modif 25/07
arrival <= '1';
else
arrival <= '0';
end if;
end process;

end structural;

--*****
--Microcontroler
--*****

library ieee;
use ieee.math_real.all;
use ieee.std_logic_1164.all;
use ieee.electrical_systems.all;
use work.new_type.all;
use work.all;

entity microcontroler is

generic ( x,y : integer; -- Microcontroler's Coordinates. -- ajout 26/07
          des: real;      -- coordinates of the destination.      -- ajout 31/07
          tol: real      -- Tolerance position destination.      -- ajout 31/07
);

port (
quantity position: in real;          -- ajout 31/07
n1,n2  : in integer;
st1,st2 : in bit; -- Cell's state, from "Interface";
arrival : in bit; -- From "Microsensor";
enable : out bit; -- For "Microactuator";
direction : out std_logic_vector(1 downto 0)-- For "Microactuator";
);
end entity microcontroler;

architecture structural of microcontroler is
quantity dod: real;
begin
dod == abs(des-position); -- Distance between the object and the destination.

process
begin
wait on arrival,st1,st2,n1,n2;
if dod < tol then -- Arrive at the destination.

```

```

enable <= '0'; -- Deactivate the cell.
    elsif arrival = '1' then
if des > position then          -- Destination is on the right or top;
if st2 = '1' and x = n2 then
enable <= '1';
direction <= "01"; -- right or top.
else
enable <= '0';
direction <= "00"; -- stop.
end if;
else
if st1 = '1' then
enable <= '1';
direction <= "10"; -- left or bottom.
else
enable <= '0';
direction <= "00"; -- stop.
end if;
end if;
    else
    enable <= '0';
    end if;
end process;

end structural;

```

```

--*****
-- Interface
--*****

```

```

library ieee;
use ieee.math_real.all;
use ieee.std_logic_1164.all;
use ieee.mechanical_systems.all;
use work.new_type.all;
use work.all;

entity interface is
generic (
vair : real; -- The velocity of the airflow.

D1 : real; -- ajout 26/07
D2 : real; -- ajout 24/07
D3 : real; -- ajout 24/07
D4 : real; -- ajout 26/07
DD : real; -- The distance between the object's edge and the hole;
DH : real; -- The width of the hole;
DL : real; -- The distance between the object's edge and the hole;
DV : real; -- The height of the hole; -- ajout 19/07
DA : real; -- The distance between the hole (V); -- ajout 24/07
DB : real; -- The distance between the hole (H); -- ajout 24/07

LENGTH : real; -- The length of the object; -- ajout 19/07
WIDTH : real; -- The width of the object;
DIM : integer; -- Dimension of the convoyeur;

```



```

Dx0, Dy0 : real -- Initial Position. -- modif 19/07
);
port (
op, op_h : in en_op; -- modif 19/07, modif 25/07
dir, dir_h : in dirflux; -- modif 25/07

quantity vairx : out real; -- The horizontal velocity of the airflow;
quantity vairy : out real; -- The vertical velocity of the airflow; -- ajout 25/07
quantity position_x : in real; -- The object's position -- X axis.
quantity position_y : in real; -- The object's position -- Y axis. -- ajout 25/07
quantity vox : in real; -- The object's velocity -- X axis.
quantity voy : in real; -- The object's velocity -- Y axis. -- ajout 25/07

quantity sband : out real; -- The cell's opening's surface covered by the object, V cell.
quantity sband_h : out real; -- The cell's opening's surface covered by the object, H cell. -- ajout 26/07
signal n1_o, n2_o, n3_o, n4_o : out integer; -- The covered microactuator's number. -- ajout 19/07
signal n1_ho, n2_ho, n3_ho, n4_ho : out integer; -- The covered microactuator's number. -- ajout 19/07
signal st1, st2 : out bit; -- The states of the object. -- Vertical cells
signal st1_h, st2_h : out bit -- The states of the object. -- Horizontal cells -- ajout 25/07
);

end entity interface;

architecture structural of interface is

quantity position_left : real; -- The position of the object's left edge;
quantity position_right : real; -- The position of the object's right edge;
quantity position_bottom : real; -- The position of the object's bottom edge; -- ajout 25/07
quantity position_top : real; -- The position of the object's top edge; -- ajout 25/07
quantity sband1, sband2, sband3 : real; -- The surface of the object's exposed to the airflow, V cell;
quantity sband1_h, sband2_h, sband3_h : real; -- The surface of the object's exposed to the airflow, H cell; -- ajout 25/07

signal direction_x, direction_y : std_logic_vector(1 downto 0) := "00"; -- The flux direction;
signal a1, a2, a3, a4, a5, a6 : boolean;
signal b1, b2, b3, b4, b5, b6 : boolean; -- ajout 26/07
signal Voz_s_z, Vox_s_z, vairx_s_z : boolean;
signal Voy_s_z, vairy_s_z : boolean; -- ajout 26/07

type states is (stage1, stage2);
signal state1 : states; -- The object's right edge's state;
signal state2 : states; -- The object's left edge's state.
signal state1_h : states; -- The object's top edge's state; -- ajout 26/07
signal state2_h : states; -- The object's bottom edge's state. -- ajout 26/07

signal n1, n2, n3, n4 : integer; -- The covered microactuator's number. -- ajout 19/07
signal n1_h, n2_h, n3_h, n4_h : integer; -- The covered microactuator's number. -- ajout 19/07

-- Initialisation of the opening's position - vertical cell.
constant left : real_vector := (0.0, D3,
D3+1.0*(DD+DH), D3+2.0*(DD+DH), D3+3.0*(DD+DH),
D3+4.0*(DD+DH), D3+5.0*(DD+DH), D3+6.0*(DD+DH),
D3+7.0*(DD+DH), D3+8.0*(DD+DH), D3+9.0*(DD+DH),
D3+10.0*(DD+DH), D3+11.0*(DD+DH), D3+12.0*(DD+DH),
D3+13.0*(DD+DH), D3+14.0*(DD+DH), D3+15.0*(DD+DH),
D3+16.0*(DD+DH), D3+17.0*(DD+DH), D3+18.0*(DD+DH),
D3+19.0*(DD+DH));

```

```

constant right: real_vector := (0.0,
D3+1.0*(DD+DH)-DD,D3+2.0*(DD+DH)-DD,D3+3.0*(DD+DH)-DD,
D3+4.0*(DD+DH)-DD,D3+5.0*(DD+DH)-DD,D3+6.0*(DD+DH)-DD,
D3+7.0*(DD+DH)-DD,D3+8.0*(DD+DH)-DD,D3+9.0*(DD+DH)-DD,D3+10.0*(DD+DH)-DD,
D3+11.0*(DD+DH)-DD,D3+12.0*(DD+DH)-DD,D3+13.0*(DD+DH)-DD,D3+14.0*(DD+DH)-DD,
D3+15.0*(DD+DH)-DD,D3+16.0*(DD+DH)-DD,D3+17.0*(DD+DH)-DD,D3+18.0*(DD+DH)-DD,
D3+19.0*(DD+DH)-DD);

constant bottom : real_vector := (0.0,D1,
D1+1.0*(DA+DV),D1+2.0*(DA+DV),D1+3.0*(DA+DV),
D1+4.0*(DA+DV),D1+5.0*(DA+DV),D1+6.0*(DA+DV),
D1+7.0*(DA+DV),D1+8.0*(DA+DV),D1+9.0*(DA+DV),
D1+10.0*(DA+DV),D1+11.0*(DA+DV),D1+12.0*(DA+DV),
D1+13.0*(DA+DV),D1+14.0*(DA+DV),D1+15.0*(DA+DV),
D1+16.0*(DA+DV),D1+17.0*(DA+DV),D1+18.0*(DA+DV),
D1+19.0*(DA+DV));

constant top: real_vector := (0.0,
D1+1.0*(DA+DV)-DA,D1+2.0*(DA+DV)-DA,D1+3.0*(DA+DV)-DA,
D1+4.0*(DA+DV)-DA,D1+5.0*(DA+DV)-DA,D1+6.0*(DA+DV)-DA,
D1+7.0*(DA+DV)-DA,D1+8.0*(DA+DV)-DA,D1+9.0*(DA+DV)-DA,D1+10.0*(DA+DV)-DA,
D1+11.0*(DA+DV)-DA,D1+12.0*(DA+DV)-DA,D1+13.0*(DA+DV)-DA,D1+14.0*(DA+DV)-DA,
D1+15.0*(DA+DV)-DA,D1+16.0*(DA+DV)-DA,D1+17.0*(DA+DV)-DA,D1+18.0*(DA+DV)-DA,
D1+19.0*(DA+DV)-DA);

-- Initialisation of the opening's position - horizontal cell.
constant left_h : real_vector := (0.0,D4,
D4+1.0*(DV+DL),D4+2.0*(DV+DL),D4+3.0*(DV+DL),
D4+4.0*(DV+DL),D4+5.0*(DV+DL),D4+6.0*(DV+DL),
D4+7.0*(DV+DL),D4+8.0*(DV+DL),D4+9.0*(DV+DL),
D4+10.0*(DV+DL),D4+11.0*(DV+DL),D4+12.0*(DV+DL),
D4+13.0*(DV+DL),D4+14.0*(DV+DL),D4+15.0*(DV+DL),
D4+16.0*(DV+DL),D4+17.0*(DV+DL),D4+18.0*(DV+DL),
D4+19.0*(DV+DL));

constant right_h: real_vector := (0.0,
D4+1.0*(DV+DL)-DL,D4+2.0*(DV+DL)-DL,D4+3.0*(DV+DL)-DL,
D4+4.0*(DV+DL)-DL,D4+5.0*(DV+DL)-DL,D4+6.0*(DV+DL)-DL,
D4+7.0*(DV+DL)-DL,D4+8.0*(DV+DL)-DL,D4+9.0*(DV+DL)-DL,D4+10.0*(DV+DL)-DL,
D4+11.0*(DV+DL)-DL,D4+12.0*(DV+DL)-DL,D4+13.0*(DV+DL)-DL,D4+14.0*(DV+DL)-DL,
D4+15.0*(DV+DL)-DL,D4+16.0*(DV+DL)-DL,D4+17.0*(DV+DL)-DL,D4+18.0*(DV+DL)-DL,
D4+19.0*(DV+DL)-DL);

constant bottom_h : real_vector := (0.0,D2,
D2+1.0*(DB+DH),D2+2.0*(DB+DH),D2+3.0*(DB+DH),
D2+4.0*(DB+DH),D2+5.0*(DB+DH),D2+6.0*(DB+DH),
D2+7.0*(DB+DH),D2+8.0*(DB+DH),D2+9.0*(DB+DH),
D2+10.0*(DB+DH),D2+11.0*(DB+DH),D2+12.0*(DB+DH),
D2+13.0*(DB+DH),D2+14.0*(DB+DH),D2+15.0*(DB+DH),
D2+16.0*(DB+DH),D2+17.0*(DB+DH),D2+18.0*(DB+DH),
D2+19.0*(DB+DH));

constant top_h: real_vector := (0.0,
D2+1.0*(DB+DH)-DB,D2+2.0*(DB+DH)-DB,D2+3.0*(DB+DH)-DB,

```

```

D2+4.0*(DB+DH)-DB,D2+5.0*(DB+DH)-DB,D2+6.0*(DB+DH)-DB,
D2+7.0*(DB+DH)-DB,D2+8.0*(DB+DH)-DB,D2+9.0*(DB+DH)-DB,D2+10.0*(DB+DH)-DB,
D2+11.0*(DB+DH)-DB,D2+12.0*(DB+DH)-DB,D2+13.0*(DB+DH)-DB,D2+14.0*(DB+DH)-DB,
D2+15.0*(DB+DH)-DB,D2+16.0*(DB+DH)-DB,D2+17.0*(DB+DH)-DB,D2+18.0*(DB+DH)-DB,
D2+19.0*(DB+DH)-DB);

```

```

-- This function is to set the initials state of the object's edges
-- according to the initial position of the object.
-- where len : length or width of the object,
-- d : the object's initial position on axis X or Y.
-- n = 1 : right or bottom edge,
-- n = 2 : left or top edge.

```

```

function state(len,d:real;left,right:real_vector;n:integer) return states is
variable i: integer;

```

```

begin
if n = 1 then -- Right edge (vertical cell)
i := integer((d-D3)/(DD+DH)+0.5);
if d>left(i) and d<right(i)
then return stage2;
else return stage1;
end if;
elsif n = 2 then -- Left edge (vertical cell)
i := integer((d-len-D3)/(DD+DH)+0.5);
if (d-len) > left(i) and (d-len) <right(i)
then return stage2;
else return stage1;
end if;
elsif n = 3 then -- Top edge (horizontal cell)
i := integer((d-D2)/(DB+DH)+0.5);
if (d-len) > left(i) and (d-len) <right(i)
then return stage2;
else return stage1;
end if;
else -- Bottom edge (horizontal cell)
i := integer((d-len-D2)/(DB+DH)+0.5);
if (d-len) > left(i) and (d-len) <right(i)
then return stage2;
else return stage1;
end if;
end if;
end function;

```

```

begin

```

```

-- Vertical cells
process -- Direction signal traitement.
begin
wait on op,dir;
for i in 1 to DIM loop
for j in 1 to DIM loop -- ajout 19/07
if op(i)(j) = '1' and dir(i)(j) = "01" then
direction_x <= "01"; -- Enable, right;
wait until op(i)(j) = '0'; -- ajout 24/07
elsif op(i)(j) = '1' and dir(i)(j) = "10" then
direction_x <= "10"; -- Enable, left;

```

```

wait until op(i)(j) = '0'; -- ajout 24/07
else
direction_x <= "00";
end if;
end loop;
end loop;

end process;

if direction_x = "01" use -- Active, right.
vairx == vair*sin(math_pi/4.0);
elsif direction_x = "10" use -- Active, left.
vairx == -vair*sin(math_pi/4.0);
else
vairx == 0.0; -- No active.
end use;

-- Horizontal cells, ajout 25/07;
process -- Direction signal traitement.
begin
wait on op_h,dir_h;
for i in 1 to DIM loop
for j in 1 to DIM loop
if op_h(i)(j) = '1' and dir_h(i)(j) = "01" then
direction_y <= "01"; -- Enable, top;
wait until op_h(i)(j) = '0';
elsif op_h(i)(j) = '1' and dir_h(i)(j) = "10" then
direction_y <= "10"; -- Enable, bottom;
wait until op_h(i)(j) = '0';
else
direction_y <= "00";
end if;
end loop;
end loop;
end process;

if direction_y = "01" use -- Active, top.
vairy == vair*sin(math_pi/4.0);
elsif direction_y = "10" use -- Active, bottom.
vairy == -vair*sin(math_pi/4.0);
else
vairy == 0.0; -- No active.
end use;

-- The right and the left edge's position.
position_right == position_x;
position_left == position_x-LENGTH;
position_top == position_y; -- ajout 25/07
position_bottom == position_y-WIDTH; -- ajout 25/07

-- The following signals are used to detect
-- the moment when the object pass the limit

```

```

-- of the cell's opening.
b6 <= position_left'above(left(n2+1));
b5 <= position_left'above(right(n2));
b4 <= position_left'above(left(n2));
b3 <= position_right'above(left(n1+1));
b2 <= position_right'above(right(n1));
b1 <= position_right'above(left(n1));
break on b1,b2,b3,b4,b5,b6;

a6 <= position_bottom'above(bottom_h(n2_h+1)); -- ajout 25/07
a5 <= position_bottom'above(top_h(n2_h)); -- ajout 25/07
a4 <= position_bottom'above(bottom_h(n2_h)); -- ajout 25/07
a3 <= position_top'above(bottom_h(n1_h+1)); -- ajout 25/07
a2 <= position_top'above(top_h(n1_h)); -- ajout 25/07
a1 <= position_top'above(bottom_h(n1_h)); -- ajout 25/07
break on a1,a2,a3,a4,a5,a6; -- ajout 25/07

-- The first FSM, for the object's right edge.
-- The function of this FSM:
-- Change the edge's state according to its
-- position compared to the cell's opening's position.

process
variable state_1 : states := state(LENGTH,Dx0,left,right,1);
variable st_1 : bit := '0';
begin
state1 <= state_1;
st1 <= st_1;
case state_1 is
when stage1 =>
n1 <= integer((position_right-D3)/(DD+DH)+0.5); -- modif 26/07
n3 <= integer((position_top-D1)/(DV+DA)+0.5); -- Position top, V cells.-- ajout 26/07

wait until b3 or not b2;
state_1 := stage2;
st_1 := '1';
when stage2 =>
n1 <= integer((position_right-D3)/(DD+DH)+0.5); -- modif 26/07
n3 <= integer((position_top-D1)/(DV+DA)+0.5); -- Position top, V cells.-- ajout 26/07

wait until b2 or not b1;
state_1 := stage1;
st_1 := '0';
when others =>
end case;
end process;

break on state1;

if state1 = stage1 use
sband1 == 0.0; -- The surface exposed to the airflow.
else
sband1 == (position_right-left(n1))*WIDTH;
end use;

-- The second FSM, for the object's left edge.
-- The function of this FSM is the same with above.
process

```

```

variable state_2 : states := state(LENGTH,Dx0,left,right,2);
variable st_2 : bit := '0';
begin
state2 <= state_2;
st2 <= st_2;
case state_2 is
when stage1 =>
n2 <= integer((position_left-D3)/(DD+DH)+0.5); -- modif 26/07
n4 <= integer((position_bottom-D1)/(DV+DA)+0.5); -- Position bottom, V cells.-- ajout 26/07

wait until b6 or not b5;
state_2 := stage2;
st_2 := '1';
when stage2 =>
n2 <= integer((position_left-D3)/(DD+DH)+0.5); -- modif 26/07
--n4 <= integer((position_bottom-D1)/(DV+DA)+0.5); -- Position bottom, V cells.-- ajout 26/07

wait until b5 or not b4;
state_2 := stage1;
st_2 := '0';
when others =>
end case;
end process;

break on state2;

-- The third FSM, for the object's top edge. -- ajout 26/07
-- The function of this FSM is the same with above.
process
variable state_1_h : states := state(WIDTH,Dy0,bottom_h,top_h,3);
variable st_1_h : bit := '0';
begin
state1_h <= state_1_h;
st1_h <= st_1_h;
case state_1_h is
when stage1 =>
n1_h <= integer((position_top-D2)/(DB+DH)+0.5); -- modif 26/07
n3_h <= integer((position_right-D4)/(DV+DL)+0.5); -- Position top, H cells.-- ajout 26/07

wait until a3 or not a2;
state_1_h := stage2;
st_1_h := '1';
when stage2 =>
n1_h <= integer((position_top-D2)/(DB+DH)+0.5); -- modif 26/07
n3_h <= integer((position_right-D4)/(DV+DL)+0.5); -- Position top, H cells.-- ajout 26/07

wait until a2 or not a1;
state_1_h := stage1;
st_1_h := '0';
when others =>
end case;
end process;

if state1_h = stage1 use -- ajout 26/07
sband1_h == 0.0; -- The surface exposed to the airflow.
else
sband1_h == (position_top-bottom_h(n1_h))*LENGTH;
end use;

-- The fourth FSM, for the object's bottom edge. -- ajout 26/07

```

```

-- The function of this FSM is the same with above.
process
variable state_2_h : states := state(WIDTH,Dy0,bottom_h,top_h,4);
variable st_2_h : bit := '0';
begin
state2_h <= state_2_h;
st2_h <= st_2_h;
case state_2_h is
when stage1 =>
n2_h <= integer((position_bottom-D2)/(DB+DH)+0.5); -- modif 26/07
n4_h <= integer((position_left-D4)/(DV+DL)+0.5); -- Position bottom, H cells.-- ajout 26/07
wait until a6 or not a5;
state_2_h := stage2;
st_2_h := '1';
when stage2 =>
n2_h <= integer((position_bottom-D2)/(DB+DH)+0.5); -- modif 26/07
n4_h <= integer((position_left-D4)/(DV+DL)+0.5); -- Position bottom, H cells.-- ajout 26/07
wait until a5 or not a4;
state_2_h := stage1;
st_2_h := '0';
when others =>
end case;
end process;

-- For the output;
n1_o <= n1; -- ajout 26/07
n2_o <= n2; -- ajout 26/07
n3_o <= n3; -- ajout 26/07
n4_o <= n4; -- ajout 26/07
n1_ho <= n1_h; -- ajout 26/07
n2_ho <= n2_h; -- ajout 26/07
n3_ho <= n3_h; -- ajout 26/07
n4_ho <= n4_h; -- ajout 26/07

-- For vertical cells.
if state2 = stage1 use
sband2 == 0.0; -- The surface exposed to the airflow.
else
sband2 == (DH+left(n2)-position_left)*WIDTH;
end use;

-- The surface covered by the object's body.
if state1 = stage1 use
sband3 == real(n1-n2)*DH*WIDTH;
elsif n1>n2 use
sband3 == real(n1-n2-1)*DH*WIDTH;
else sband3 == 0.0;
end use;

-- The total exposed surface
sband == sband1 + sband2 + sband3;

-- For horizontal cells. -- ajout 26/07
if state2_h = stage1 use
sband2_h == 0.0; -- The surface exposed to the airflow.

```

```

else
sband2_h == (DH+bottom_h(n2_h)-position_bottom)*LENGTH;
end use;

-- The surface covered by the object's body.
if state1_h = stage1 use
sband3_h == real(n1_h-n2_h)*DH*LENGTH;
elsif n1_h>n2_h use
sband3_h == real(n1_h-n2_h-1)*DH*LENGTH;
else sband3_h == 0.0;
end use;

-- The total exposed surface
sband_h == sband1_h + sband2_h + sband3_h;

end;

--*****
-- Object
--*****

library ieee;
use ieee.math_real.all;
use ieee.std_logic_1164.all;
use ieee.mechanical_systems.all;
use ieee.electrical_systems.all;
use work.new_type.all;
use work.all;

entity object is

generic (
Rho : real := 1.3;      -- The air's density (kg/m3).
Cxf : real := 0.004; -- The air's friction parameter.
Cxp : real := 1.11; -- The air's pressure parameter.
mass : real; -- Mass of the object (kg).
LENGTH : real; -- The length of the object;
WIDTH : real; -- The width of the object;
TK : real; -- The thickness of the object.
Dx0,Dy0,Vox0,Voy0,Height0:real; -- Initial values. -- modif 26/07
voff :real -- Velocity of the airfow static.
);

port (
st1,st2,st1_h,st2_h : in bit; -- ajout 26/07
quantity vax_on : in real; -- The horizontale velocity of the airflow;
quantity vay_on : in real; -- The vertical velocity of the airflow; -- ajout 26/07
quantity sband : in real; --
quantity sband_h : in real; --
quantity position_x : out real; -- The object's position -- X axis.
quantity position_y : out real; -- The object's position -- Y axis. -- ajout 26/07
quantity vox,voy : out real -- modif 26/07
);

end entity object;

```



```

architecture structural of object is

constant g : real := 9.8;    -- Gravitation, m/s2.

-- Z axis.
quantity Dz : real; -- The object's Height;
quantity Fl : real; -- Total vertical force of the airflow;
quantity weight: real; -- The heaviness;
quantity deltap : real; -- The difference of the pressure.
quantity Fsp : real; -- The force created by the sous pressure.
quantity Flr: real; -- The vertical resistance - pressure.
quantity Voz : real; -- The object's vertical velocity.
signal fall,Voz_s_z : boolean;

-- X axis.
quantity Fcx: real; -- The horizontal drag force of the airflow.
quantity Fdpx1,Fdpx2: real; -- The horizontal drag force - pressure.
quantity Fcrx: real; -- The air's horizontal resistance.
quantity Frfx: real; -- The horizontal resistance - friction.
quantity Frpx: real; -- The horizontal resistance - pressure.
signal Vox_s_z,Vairx_s_z : boolean;

-- Y axis.
quantity Fcy: real; -- The horizontal drag force of the airflow.
quantity Fdpy1,Fdpy2: real; -- The horizontal drag force - pressure.
quantity Fcry: real; -- The air's horizontal resistance.
quantity Frfy: real; -- The horizontal resistance - friction.
quantity Frpy: real; -- The horizontal resistance - pressure.
signal Voy_s_z,Vairy_s_z : boolean;

begin

break position_x => Dx0, Vox => Vox0, Dz => Height0;
break position_y => Dy0, Voy => Voy0;

----- Z axis -----

-- Pesentor,
weight == mass*g;

-- The airflow's vertical drag force,
Fl*(1.0+Dz) == 0.5*Rho*sband*Cxp*voff**2;

-- The force created by the difference of the pressure.
deltap == 0.5*Rho*vax_on**2;
Fsp == deltap*WIDTH*LENGTH;

-- The object rebounds when hits the floor,
break Voz => -Voz when fall;
fall <= not Dz'above(0.0);
break on fall;

-- Calculation of the air's resistance.
Voz_s_z <= Voz'above(0.0);
break on Voz_s_z;
if Voz_s_z use

```

```

    Flr == 0.5*Rho*WIDTH*LENGTH*Cxp*Voz**2;
else
Flr == -0.5*Rho*WIDTH*LENGTH*Cxp*Voz**2;
end use;

Fl-Weight-Flr-Fsp == mass*Voz'dot;
Voz == Dz'dot;

----- X axis -----

if st1 = '0' use
Fdp1 == 0.0;
elsif not Vairx_s_z use
Fdp1 == -0.5*Rho*WIDTH*TK*Cxp*(vax_on-Vox)**2;
else
Fdp1 == 0.0;
end use;

if st2 = '0' use
Fdp2 == 0.0;
elsif Vairx_s_z use
Fdp2 == 0.5*Rho*WIDTH*TK*Cxp*(vax_on-Vox)**2;
else
Fdp2 == 0.0;
end use;

-- Airflow's drag force.
Vairx_s_z <= vax_on'above(0.0);
break on vairx_s_z;
if vairx_s_z use
--Fdx == Fdfx + Fdp2;
Fcx == Fdp2;
else
--Fdx == -Fdfx + Fdp1;
Fcx == Fdp1;
end use;

-- Air's resistance.
Vox_s_z <= Vox'above(0.0);

break on Vox_s_z;
Fcrx == Frfx + Frpx;

Frfx == 0.5*Rho*WIDTH*LENGTH*Cxf*Vox**2;
Frpx == 0.5*Rho*WIDTH*TK*Cxp*Vox**2;

-- The object's dynamic;
Fcx - Fcrx == mass*Vox'dot;
Vox == position_x'dot;

----- Y axis -----

if st1_h = '0' use
Fdpi1 == 0.0;
elsif not Vairy_s_z use
Fdpi1 == -0.5*Rho*LENGTH*TK*Cxp*(vay_on-Voy)**2;

```

```

else
Fdpi1 == 0.0;
end use;

if st2_h = '0' use
Fdpi2 == 0.0;
elsif Vairy_s_z use
Fdpi2 == 0.5*Rho*LENGTH*TK*Cxp*(vay_on-Voy)**2;
else
Fdpi2 == 0.0;
end use;

-- Airflow's drag force.
Vairy_s_z <= vay_on'above(0.0);
break on vairy_s_z;
if vairy_s_z use
Fcy == Fdpi2;
else
Fcy == Fdpi1;
end use;

-- Air's resistance.
Voy_s_z <= Voy'above(0.0);
break on Voy_s_z;

Fcry == Frfy + Frpy;

Frfy == 0.5*Rho*WIDTH*LENGTH*Cxf*Voy**2;
Frpy == 0.5*Rho*LENGTH*TK*Cxp*Voy**2;

-- The object's dynamic;
Fcy - Fcry == mass*Voy'dot;
Voy == position_y'dot;

end;

--*****
-- System
--*****

library ieee;
use ieee.math_real.all;
use ieee.std_logic_1164.all;
use ieee.mechanical_systems.all;
use ieee.electrical_systems.all;
use work.new_type.all;
use work.all;

entity system is
end entity system;

architecture structural of system is

component valve
generic (m,kv,rd,st,epsilon,e_young,lv,h,g0,w,vi:real);
port (

```

```

enable : in bit; -- From "Microcontrôler".
direction: in std_logic_vector(1 downto 0);
op : out bit
);
end component;

component microsenseur
generic ( x,y : integer );
port (
          n1,n2,n3,n4 : in integer; -- From "Interface";
arrival : out bit -- For "Microactuateur";
);
end component;

component microcontrôler
generic (x,y : integer; des,tol : real);
port (
          position: in real;          -- ajout 31/07
          n1,n2   : in integer;
st1,st2 : in bit;
arrival : in bit;          -- From "Microsenseur";
enable : out bit; -- For "Microactuateur";
direction : out std_logic_vector(1 downto 0)-- For "Microactuateur";
);
end component;

constant DIM : integer := 20;          -- Dimension of the convoyeur;
constant CELL : real := 0.001; -- Cell's dimension; -- ajout 24/07
constant WIDTH : real := 0.0045; -- Object's dimension; -- ajout 24/07
constant LENGTH : real := 0.0045; -- ajout 24/07
constant TK : real := 0.00025; -- ajout 24/07
constant mass : real := 0.0000066; -- ajout 24/07

constant DH : real := 0.00013; -- Opening's dimension; -- ajout 24/07
constant DL : real := 0.00013; -- ajout 24/07
constant DD : real := CELL-DH; -- ajout 24/07
constant DV : real := CELL-DL; -- ajout 24/07
constant DA : real := DL; -- ajout 26/07 modif 18/09
constant DB : real := DD; -- ajout 26/07 modif 18/09

constant D1 : real := DL; -- ajout 26/07
constant D2 : real := D1+(DV-DH)/2.0; -- ajout 26/07 modif 18/09
constant D3 : real := DD; -- ajout 26/07
constant D4 : real := D3-(DV-DH)/2.0; -- ajout 26/07

constant Vin : real := 140.0; -- Supply voltage;

constant Dx0:real:= 0.005;
constant Dy0:real:= 0.005; -- ajout 19/07
constant Vox0:real:=0.005;
constant Voy0:real:=0.005; -- ajout 26/07
constant Height0:real:=0.05;
constant voff : real := 2.0; -- ajout 24/07
constant vair : real := 5.0;

constant des_x : real := 0.01; -- destination, -- ajout 31/07

```

```

        constant des_y : real := 0.01;    -- destination,          -- ajout 31/07
        constant tol : real := 0.0005;    -- destination,          -- ajout 31/07

signal n1,n2,n3,n4 : integer; -- From "Sensor", the cell's number,    ajout 19/07
signal n1_h,n2_h,n3_h,n4_h : integer; -- From "Sensor", the cell's number,    ajout 26/07
signal st1,st2: bit;
signal st1_h,st2_h: bit; -- ajout 26/07

signal arr,arr_h : en_op; -- modif 19/07  modif 25/07
signal en,en_h : en_op;      -- modif 19/07  modif 25/07
signal op,op_h : en_op;      -- modif 19/07  modif 25/07

signal dir : dirflux;        -- The flux direction;
signal dir_h : dirflux; -- The flux direction of the vertical cells; -- ajout 25/07

quantity position_x,vox : real;
quantity position_y,voy : real;      -- ajout 25/07
quantity sband,sband_h : real;
quantity vax_on,vay_on: real;

begin

    -- For vertical cells;
gen_valv1 : for i in 1 to DIM generate
gen_valv2 : for j in 1 to DIM generate
val: valve generic map (m=>6.61E-9, kv=>1.81E-5, rd=>0.8,
st=>1.5E-5, epsilon=>8.85E-12, e_young=>1.3E+11,
lv=>9.0E-4, h=>1.0E-4, g0=>2.0E-5,w=>1.0E-5,vi=>Vin)
port map (enable=>en(i)(j),direction=>dir(i)(j),op=>op(i)(j));
end generate;
end generate;

gen_sen1 : for i in 1 to DIM generate
gen_sen2 : for j in 1 to DIM generate
    sen : microsensor generic map (x=>i, y=>j)
port map (n1,n2,n3,n4,arr(i)(j));
end generate;
end generate;

gen_con1 : for i in 1 to DIM generate
gen_con2 : for j in 1 to DIM generate
control : microcontroler generic map (x=>i,y=>j,des=>des_x,tol=>tol)
port map (position_x,n1,n2,st1,st2,arr(i)(j),en(i)(j),dir(i)(j));
end generate;
end generate;

    -- For horizontal cells;
gen_valv3 : for i in 1 to DIM generate
gen_valv4 : for j in 1 to DIM generate
val_h: valve generic map (m=>6.61E-9, kv=>1.81E-5, rd=>0.8,
st=>1.5E-5, epsilon=>8.85E-12, e_young=>1.3E+11,
lv=>9.0E-4, h=>1.0E-4, g0=>2.0E-5,w=>1.0E-5,vi=>Vin)
port map (enable=>en_h(i)(j),direction=>dir_h(i)(j),op=>op_h(i)(j));
end generate;
end generate;

```

```

gen_sen3 : for i in 1 to DIM generate
gen_sen4 : for j in 1 to DIM generate
    sen_h : microsensor generic map (x=>i, y=>j)
port map (n1_h,n2_h,n3_h,n4_h,arr_h(i)(j));
end generate;
end generate;

gen_con3 : for i in 1 to DIM generate
gen_con4 : for j in 1 to DIM generate
control_h : microcontroller generic map (x=>i, y=>j,des=>des_y,tol=>tol)
port map (position_y,n1_h,n2_h,st1_h,st2_h,arr_h(i)(j),en_h(i)(j),dir_h(i)(j));
end generate;
end generate;

-- For the system;
inter : entity interface generic map (vair,D1=>D1,D2=>D2,D3=>D3,D4=>D4,
DD=>DD,DH=>DH,DL=>DL,DV=>DV,DA=>DA,DB=>DB,
LENGTH=>LENGTH,WIDTH=>WIDTH,DIM=>DIM,Dx0=>Dx0,Dy0=>Dy0) -- modif 20/07
port map (op,op_h,dir,dir_h,vax_on,vay_on,position_x,position_y,vox,voy,sband,
sband_h,n1,n2,n3,n4,n1_h,n2_h,n3_h,n4_h,st1,st2,st1_h,st2_h); -- modif 26/07

obj : entity object generic map (Rho=>1.3,Cxf=>0.004,Cxp=>1.11,
mass=>mass,LENGTH=>LENGTH,WIDTH=>WIDTH,TK=>TK,
Dx0=>Dx0,Dy0=>Dy0,Vox0=>Vox0,Voy0=>Voy0,Height0=>Height0,voff=>voff) -- modif 26/07
port map (st1=>st1,st2=>st2,st1_h=>st1_h,st2_h=>st2_h,-- modif 26/07
vax_on=>vax_on,vay_on=>vay_on,sband=>sband,sband_h=>sband_h,
position_x=>position_x,position_y=>position_y,vox=>vox,voy=>voy);

end;

```

Annexe B

Études de la micro-valve : La tension $V_{pull-in}$: comparaison et analyse

B.1 Les formules de $V_{pull-in}$ pour les trois formes

- **Forme 1**

$$V_{pull-in} = \sqrt{\frac{8g_0^3 k_{eq}}{27\epsilon_0 L_v h}} \quad (\text{B.1})$$

- **Forme 2**

$$V_{pull-in} = \sqrt{\frac{8g_0^2 (g_0 + 3htan\theta) k_{eq}}{27\epsilon_0 L_v h}} \quad (\text{B.2})$$

- **Forme 3**

$$V_{pull-in} = \sqrt{\frac{8(g_0 + 2\Delta w)^2 (g_0 + 2\Delta w + 3htan\theta) k_{eq}}{27\epsilon_0 L_v h}} \quad (\text{B.3})$$

B.2 Les résultats de simulation

Les figures suivantes montrent les différentes tensions $V_{pull-in}$ obtenues respectivement par les équations dynamiques du modèle et la formule théorique.

La première courbe représente la tension que l'on applique sur le modèle, elle augmente progressivement.

La deuxième courbe représente le mouvement de la micro-valve. Nous pouvons constater qu'il y a un saut dans le mouvement de la micro-valve. Le saut correspond à l'ouverture de la micro-valve, et la tension qui provoque l'ouverture brutale de la micro-valve est la tension $V_{pull-in}$.

La troisième courbe est une constante, elle représente la tension $V_{pull-in}$ de théorie calculée à partir de la formule.

B.2.1 Forme 1

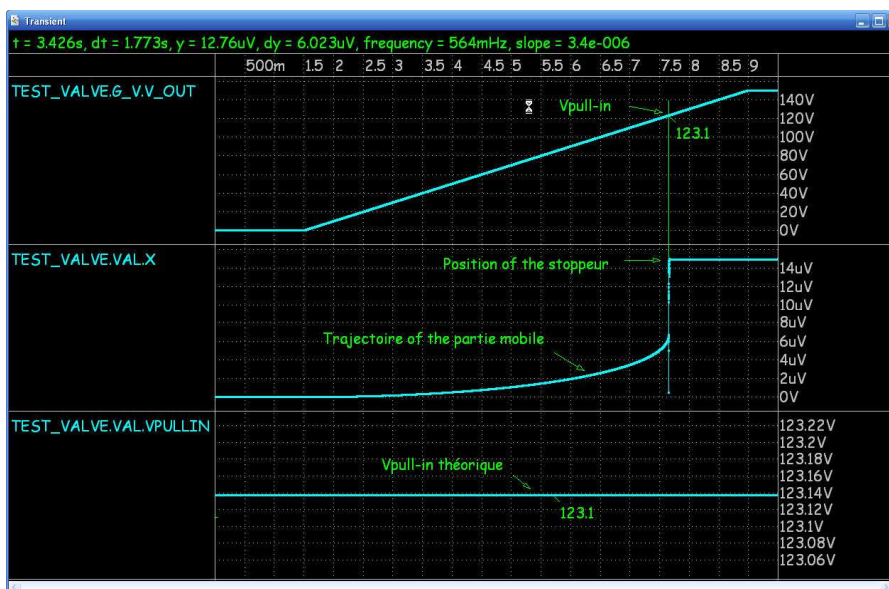


Fig. B.1 – Résultat de simulation (1) - Forme 1

B.2.2 Forme 2

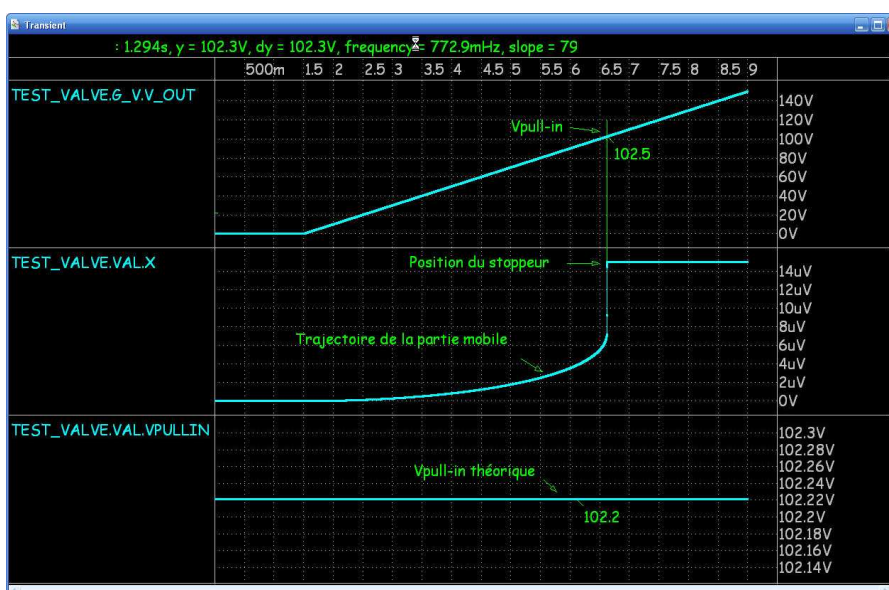


Fig. B.2 – Résultat de simulation (2) - Forme 2

B.2.3 Forme 3

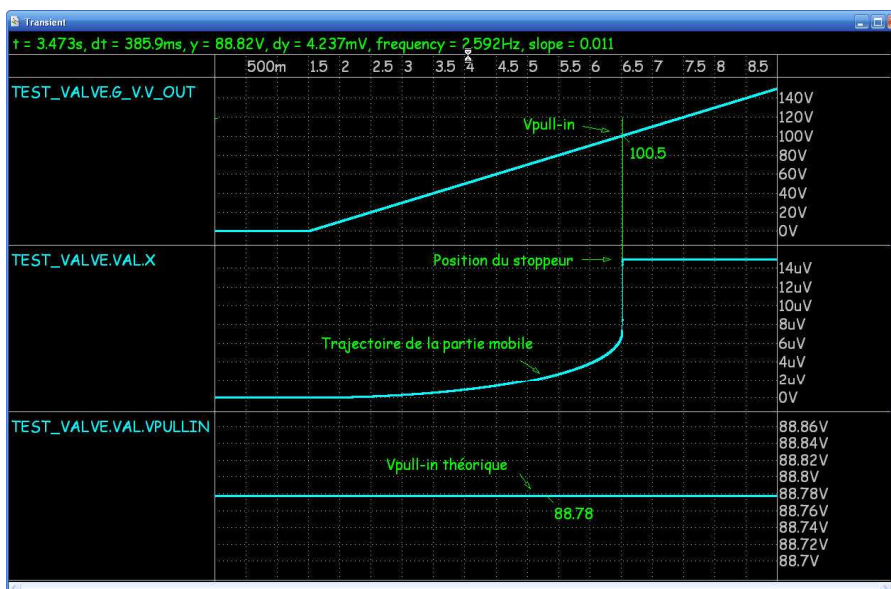


Fig. B.3 – Résultat de simulation (3) - Forme 3 : $\Delta w = 0.1 \mu m$

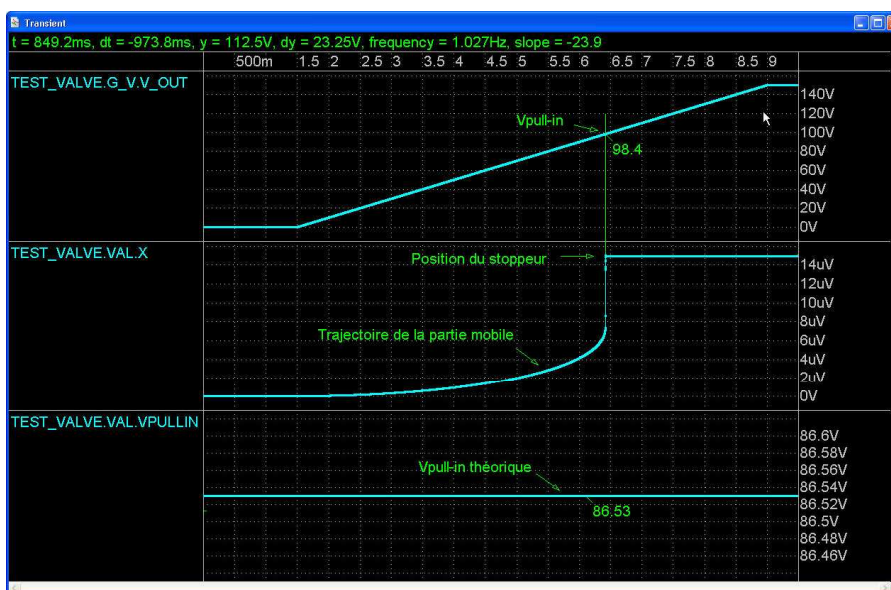


Fig. B.4 – Résultat de simulation (4) - Forme 3 : $\Delta w = 0.2 \mu m$

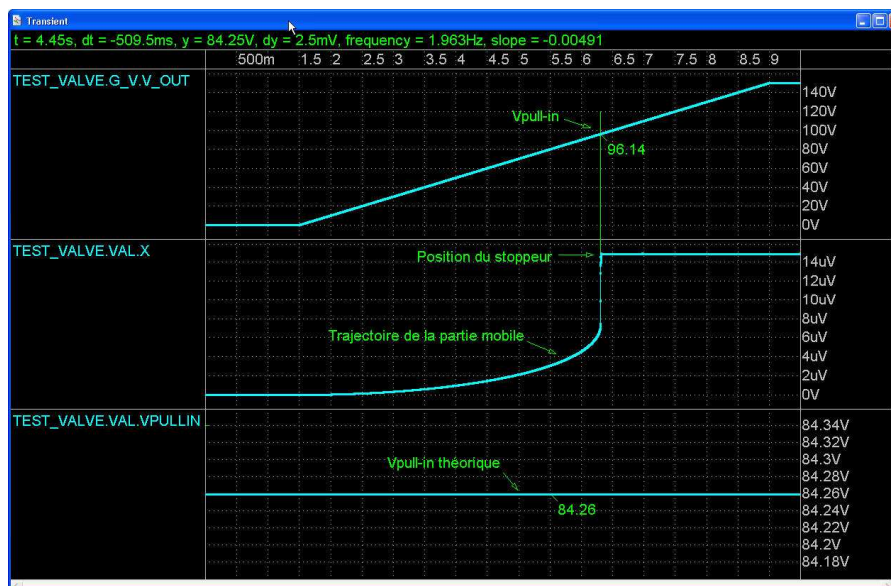


Fig. B.5 – Résultat de simulation (5) - Forme 3 : $\Delta w = 0.3 \mu m$

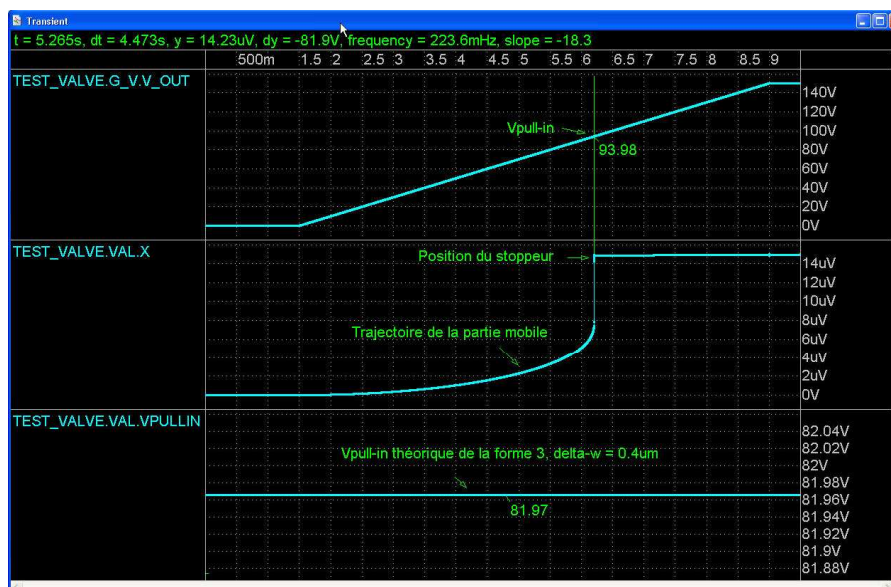


Fig. B.6 – Résultat de simulation (6) - Forme 3 : $\Delta w = 0.4 \mu m$

B.3 Analyse des résultats

B.3.1 Le bilan récapitulatif des résultats

Le tableau suivant récapitule les résultats obtenus par le modèle, les résultats obtenus par la formule et les résultats mesurés.

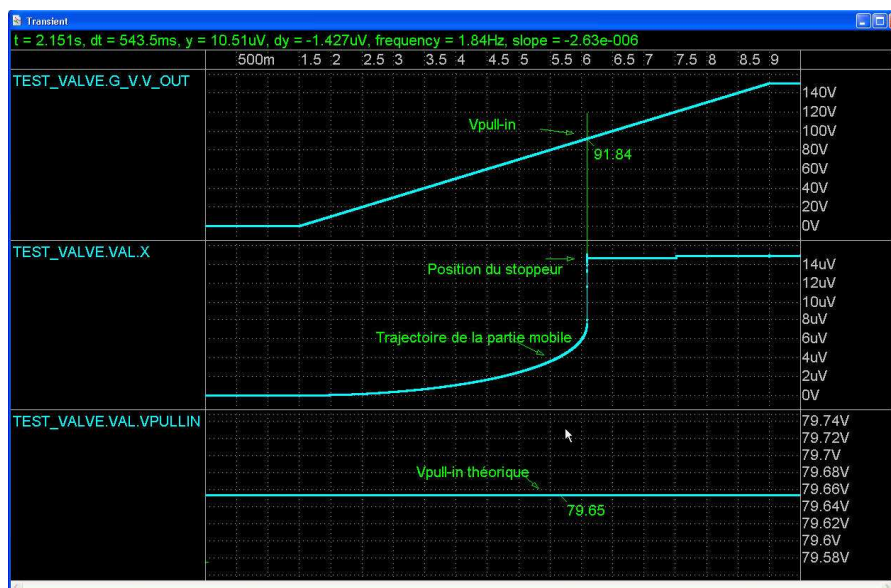


Fig. B.7 – Résultat de simulation (7) - Forme 3 : $\Delta w = 0.5 \mu m$

	$\Delta w(\mu m)$	$V_{pull-in}$ mesurée (V)	$V_{pull-in}$ modèle (V)	$V_{pull-in}$ théorique (V)	V mesurée / V modèle
Forme 1			123,1	123,1	
Forme 2			102,5	102,2	
Forme 3	0,1	103,5	100,5	88,78	1,029
	0,2	102,0	98,4	86,53	1,037
	0,3	99,0	96,14	84,26	1,030
	0,4	97,5	93,98	81,94	1,037
	0,5	95,5	91,84	79,65	1,040

Tab. B.1 – Comparaison des résultats

B.3.2 Analyse

Pour la forme 1 et la forme 2, on constate que la tension $V_{pull-in}$ obtenue par les équations dynamiques (VD) est presque identique que la tension $V_{pull-in}$ obtenue par la formule (VF). Cela signifie que les équations dynamiques du modèle reproduit fidèlement le résultat théorique. Pour l'instant, il nous manque des résultats expérimentaux (VE) pour vérifier si ces résultats théoriques correspondent ou non à la réalité.

Alors que pour la forme 3, on constate que les tensions VF sont très différentes de VD (Il est possible que la formule de $V_{pull-in}$ pour la forme 3 présente une faute). Pourtant, les résultats VF sont très proches de VE (Voir le tableau 1). La différence entre les deux résultats est moins de 5%. Cette similitude s'avère fidèle grâce à 5 résultats indépendants qui sont mesurés sur 5 situations différentes ($\Delta w = 0,1, 0,2, 0,3, 0,4, \text{ et } 0,5 \mu m$).

