REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'enseignement supérieur et de la recherche scientifique

Université de Jijel
Faculté des Sciences et de la Technologie
Département d'Electronique

Matière : ***MIC 30***

Intitulée : # Simulation des composants Semi-conducteurs

Master 2 - Microélectronique

Dr. Fatiha BOUAZIZ

Sommaire

Chapitre III : Les langages de description des systèmes mixtes

Chapitre IV : Conception des systèmes microélectroniques analogiques-numériques via VHDL-AMS

Introduction générale

Ce polycopié de cours est destiné aux étudiants de ***Master 2*** Electronique, spécialité ***Microélectronique*** dont l'intitulé est : ''**Simulation des Composants Semi-conducteurs**'. Ce cours est élaboré selon le programme officiellement agréé et confirmé par le **CPNDST,** contenant quatre chapitres et des références bibliographiques.

La simulation représente une tâche primordiale dans le chemin de la conception des systèmes électroniques. En effet, elle permet de corriger plus facilement les erreurs éventuelles et d'optimiser le coût de développement et d'industrialisation avant d'entamer les démarches de matérialisation et de réalisation technologiques. En outre, elle permet aussi d'envisager des scénarios non mesurables sur des composants réels sans les détruire.

Depuis plusieurs années, les systèmes électroniques deviennent des systèmes hétérogènes mettant en œuvre à la fois l'électronique numérique (mémoires, compteurs,..), l'électronique analogique (transistors, diodes,..), l'optique, le thermique, la mécanique...etc. Par conséquent, la conception et la simulation de tels systèmes représentent un argument puissant de motivation pour le développement des nouveaux outils et logiciels informatiques permettant de gérer la complexité croissante de ces systèmes tout en assurant la fiabilité des circuits fabriqués.

Le logiciel 'VHDL-AMS' représente l'outil informatique le plus répondu pour la description matérielle de haut niveau des systèmes mixtes à cause de sa simplicité de syntaxe d'une part, et d'autre part il permet une haute modularité facilitant des descriptions hiérarchiques. En plus, ce dernier peut travailler simultanément comme un simulateur à évènements discrets et un solveur d'équations différentielles. Par conséquent, tous les systèmes pouvant être modélisés sous Matlab, VHDL et Spice peuvent aujourd'hui être modélisés sous un même langage, qui est le 'VHDL-AMS'.

Les chapitres sont répartis comme suit :

- Le premier chapitre, présente une brève description des méthodes de modélisation et de conception des systèmes analogiques et mixtes.
- Le deuxième chapitre est consacré à la présentation de divers algorithmes de modélisation des circuits analogiques. Nous présentons tout d'abord l'architecture interne d'un simulateur analogique de type 'SPICE'. Puis, les méthodes de modélisation sont décrites pour chaque type d'analyse possible en simulation électrique.

- Dans le troisième chapitre, les langages de modélisation des circuits électroniques numériques sont présentés, suivi par une description de quelques types de logiciels de modélisation des systèmes mixtes (analogiques-numériques).

- Le quatrième chapitre est destiné à la présentation de l'environnement de travail du langage VHDL-AMS dans un premier temps. Puis, une description détaillée des modèles VHDL-AMS est effectuée par étude de quelques exemples de modélisation des circuits électroniques.

Chapitre I

Modélisation et Simulation des systèmes analogiques et mixtes

1. Introduction

La tâche de conception de fonctions électroniques sur circuit intégré ne peut être aujourd'hui menée à bien sans besoin des outils informatiques, véritables plate-formes logicielles d'aide à la conception. La complexité croissante des circuits intégrés nécessite une amélioration constante des méthodes et outils de conception, afin de réduire au maximum le temps de développement, tout en assurant la fiabilité des circuits fabriqués.

Nous abordons dans ce chapitre l'aspect fondamental de la conception des systèmes analogiques et mixtes ainsi que les méthodes différentes de la modélisation hiérarchique.

2. Notions fondamentales sur la conception des systèmes

Du fait de la complexité et de l'hétérogénéité des systèmes électroniques, les concepteurs doivent gérer des projets associant plusieurs disciplines et plusieurs technologies. Ce problème pluridisciplinaire ainsi que le besoin d'optimiser le processus de conception pour réduire le temps de la mise sur le marché, nécessite de mettre en place des méthodes et des outils facilitant la création des circuits (analogiques, numériques et mixtes). En outre, les concepteurs doivent gérer la coordination de tous les aspects mis en jeu dans une conception, comme : les spécifications et les performances, les modèles, les règles de conception, les technologies, les objectifs et les contraintes, les méthodes de conception, les langages de programmation, l'automatisation, la fabrication,…etc.

2.1 Description de la conception hiérarchique

Récemment, en raison de l'hétérogénéité et de la grande complexité des systèmes intégrés sur une puce, la hiérarchisation de la conception s'avère nécessaire. Autrement dit, le concepteur commence par concevoir et valider un système à l'aide de blocs fonctionnels, puis il descend progressivement dans le détail des blocs, jusqu'à la conception de circuits élémentaires au niveau transistor ou portes logiques.

Lors de la conception d'un système, le problème initial étant la traduction du cahier des charges en un circuit intégré fonctionnel. Cette approche revient à décomposer le problème en des sous-problèmes et donnant lieu à plusieurs niveaux hiérarchiques. On distingue deux modes de conception d'un système correspondant aux deux sens de parcours de l'hiérarchie : *la conception en mode descendant (Top-Down)* et *la conception en mode ascendant* (*Bottom-Up*). Chaque niveau hiérarchique est caractérisé par un ensemble d'entités permettant de décrire la topologie du système.

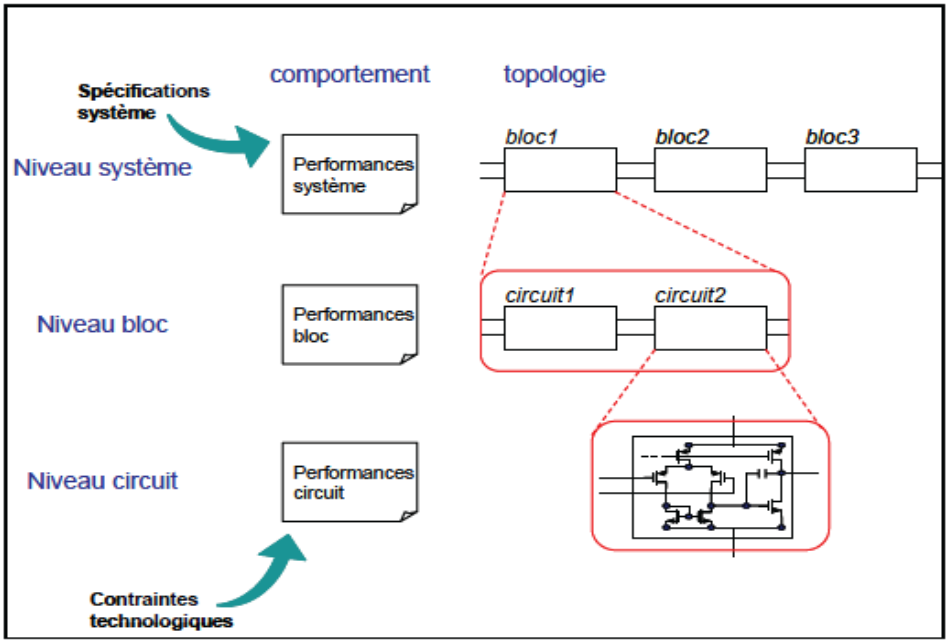


Figure I.1 **:** *Description de la conception hiérarchique.*

La figure I.1 présente une description générale de la conception hiérarchique en mode descendant. Dans ce cas, le système est décomposé en trois niveaux hiérarchiques :

- **Niveau système :** la topologie du système est décrite au moyen de blocs fonctionnels.
- **Niveau bloc :** les primitives de chaque bloc sont des circuits.
- **Niveau circuit :** les primitives à ce niveau sont des composants électroniques de base (transistors, diodes, résistances,…).

La conception hiérarchique est contrainte au plus haut niveau par les *spécifications système* ainsi qu'au plus bas niveau par les *contraintes technologiques.* Les spécifications système sont les données du client fixant les performances du système, tandis que les contraintes technologiques imposent les marges de conception.

2.2 La spécification

La spécification représente la première étape de développement d'un système (cahier des charges). Elle consiste d'une part à prendre en considération les exigences fonctionnelles et les contraintes de réalisation, et d'autre part, elle précise le schéma directeur qui sera choisi pour réaliser le système. Par conséquence, le fait de spécifier un système consiste à répondre à la question « que doit faire ce système ? ».

2.3 Flot de conception

La succession des décisions permettant de passer d'un niveau hiérarchique à un niveau adjacent est appelée *flot de conception*. Le parcours Top-Down du flot de conception consiste à propager et distribuer les spécifications système vers les niveaux inférieurs, jusqu'au schéma transistor de chaque circuit. Le parcours Bottom-Up consiste à vérifier que l'implémentation électrique permet bien de réaliser un système conforme aux spécifications initiales.

2.4 Modèle et modélisation

- Le modèle consiste essentiellement à développer une représentation abstraite d'une réalité physique. Le modèle dépend du point de vue selon lequel on observe le système, mais aussi suivant l'utilisation que l'on souhaite faire de ce modèle au sein du processus de conception.
- La modélisation représente la tâche centrale de la conception. Elle consiste à trouver une loi mathématique représentative du comportement d'un système et la vérification de la vraisemblance de cette loi se fait par comparaison avec des données de référence provenant des mesures avant tous et parfois des simulations réalisées à partir des modèles déjà validés.
- Les principaux critères de qualité d'un modèle sont la précision et la rapidité d'exécution. La précision d'un modèle dépend de sa capacité à couvrir un grand nombre de contextes d'utilisation ; plus le modèle est précis, plus le nombre d'équations et de paramètres est important. Tandis que la rapidité d'exécution dépend de la méthode d'implémentation des équations, de l'outil de simulation et aussi de la précision du modèle.

2.5 Niveaux d'abstraction

Dans le cas de la conception des systèmes analogiques, un *modèle comportemental* représente tout modèle décrivant le comportement électrique du circuit autrement que la description structurelle au niveau transistor. Lors de la conception, les modèles sont considérés à des niveaux d'abstraction différents. On distingue souvent trois niveaux d'abstraction :

- **Niveau *fonctionnel*** : définition des relations entrée/sortie et de la représentation de la fonction idéale.
- **Niveau *comportemental*** : description des caractéristiques de la réalisation physique ainsi que ses non-idéalités.
- **Niveau circuit** : description avec les primitives de plus bas niveau (transistors).

L'intérêt de ces différents niveaux d'abstraction est qu'ils peuvent être utiles à différents niveaux hiérarchiques du flot de conception donnant le meilleur compromis rapidité/ précision.

2.6 Technique de modélisation

La tâche consistant à implémenter un modèle dans un style de langage donné s'appelle *technique de modélisation*. Cette dernière dépend fortement du type de comportement observé (continu, discret, logique), ainsi que du niveau hiérarchique auquel on se place pour étudier un système.

En analogique, La validation d'un système électronique est essentiellement basée sur des logiciels de simulation électriques (le logiciel *SPICE es*t le plus connu). Ceux-ci font appel à des modèles de différents composants utilisés (transistors, diodes, résistances, capacités,...), qui en décrivent le *comportement*, c'est à dire les relations entre les signaux présents sur les points d'entrée/sortie (E/S). Ces modèles décrivent les relations macroscopiques entre tensions et courants de diverses bornes, sous forme d'équations différentielles. Il s'agit donc d'une représentation mathématique de phénomènes physiques auxquels obéissent les composants.

L'apparition des langages de description matérielle a permis de modéliser les circuits directement en implémentant des *équations différentielles algébriques* (adaptées aux comportements analogiques continus dans le temps) ou des *algorithmes dirigés par évènements* (adaptés aux comportements des circuits à temps discret). Enfin, on trouve

également une technique de modélisation ne passant pas par la recherche d'équations comportementales mais qui consiste à modéliser sous la forme de tableaux. On décrit alors les relations entrées/sorties numériquement, pour des valeurs bien précises des entrées et avec un degré de paramétrage variable.

3. Méthodologies de modélisation

La modélisation comportementale signifie le processus partant de l'analyse du comportement d'un circuit et aboutissant à un système d'équations ou un algorithme descriptif. Ce processus diffère selon les phases de la conception d'un système : en phase descendante on parle de *raffinement* des modèles, dans le sens où ils vont vers des niveaux de précision croissants (grand nombre de variables d'état et de performances modélisées), en phase montante on parle à l'inverse de *simplification* de modèles.

3.1 Modélisation en phase Bottom-Up (ascendante)

Dans la phase de conception ascendante (Bottom-Up), le point de départ est un circuit dimensionné ayant un objectif précis. Il s'agit d'en extraire un modèle comportemental, nécessairement plus abstrait que la description au niveau transistor, mais capable de propager les performances du circuit réalisé vers les haut niveaux hiérarchiques. Dans ce cas deux types de modèles sont possibles : *modèle basé sur la connaissance* et *modèle basé sur l'identification.*

a. Modèle basé sur la connaissance : cette approche est basée sur la connaissance de la structure interne du circuit (topologie) et de lois électriques simples (lois de Kirchoff, modèles du premier ordre des transistors : Ebers-Moll ou MOS niveau 1) et Il s'agit d'effectuer un calcul symbolique et d'exprimer les performances du circuit à partir des paramètres clés de ses composants. On aboutit à un système d'équations qui pourra éventuellement être simplifié. Le principal avantage est que le modèle ainsi développé garde un sens physique et peut même être prédictif, si le domaine de validité du système est clairement connu.

Les inconvénients de cette approche sont que l'obtention du système d'équations est une tâche qui devient rapidement complexe avec l'augmentation de la taille du circuit et qu'elle est plus facilement applicable aux circuits linéaires.

b. Modèle basé sur l'identification : Il s'appuie sur l'acquisition des caractéristiques externes du circuit, sans tenir compte de sa structure interne. Les données de caractérisation se présentent donc sous forme tabulaire et il s'agit de trouver une expression mathématique reproduisant le bon comportement, par une méthode d'interpolation.

Le principal avantage de cette méthode est qu'elle présente un grand potentiel d'automatisation. Tandis que, l'inconvénient major de ce modèle est que ses paramètres perdent leur sens physique autrement que la méthode symbolique précédente basée sur la connaissance.

3.2 Modélisation en phase Top-Down (descendante)

Dans cette phase de conception, le schéma électrique étant inconnu et le point de départ est une liste de spécifications. L'objectif est alors de construire un modèle décrivant la fonction du bloc considéré ajustable par des paramètres de performances.

Le modèle en phase descendante est souvent utilisé en amont de la tâche de conception pour poser les bases de l'architecture du système et la valider par une première série de simulations. Par conséquence, les spécifications du système sont distribuées sur chaque bloc.
Les modèles extraits en phase Top-Down sont des modèles *fonctionnels* caractérisés par une précision moindre que celle des modèles *comportementaux* extraits en phase Bottom-Up. Tandis que la rapidité est prédominante dans cette phase.

4. Simulation des systèmes mixtes (analogique-digitale)

La simulation mixte permet d'étudier le comportement temporel de systèmes complexes en un temps extrêmement réduit par rapport à une simulation uniquement électrique. Ce type de simulation est en effet basé sur l'abstraction de la partie digitale à un niveau fonctionnel logique. Pour cette partie, les grandeurs étudiées ne sont donc plus électriques mais numériques et sont caractérisées par leurs changements d'état. Des algorithmes *dirigés par évènements (event-driven)* permettent d'étudier de manière très efficace l'évolution des signaux digitaux. Une simulation mixte peut être décomposée en trois différentes phases:

4.1 Phase d'élaboration

Elle correspond à la décomposition du circuit mixte en blocs distincts analogiques et digitaux, chaque partie étant traitée par les algorithmes concernés. Aux interfaces entre les deux parties, doivent être placés des modèles plus ou moins élaborés de *convertisseurs* A/D et

D/A, qui assurent la correspondance des données entre les algorithmes analogiques et digitaux.

4.2 Phase d'initialisation

Il s'agit de déterminer le point de fonctionnement du système, c'est à dire l'état initial de toutes les grandeurs mises en jeu (tensions, courants, états logiques). Cette recherche est indispensable au simulateur analogique et correspond à une analyse DC. Pour la partie digitale, cette notion dépend du simulateur: cela peut correspondre soit à une initialisation (solution au temps 0), soit à une certaine durée, appelée temps de *setup*, au bout de laquelle un état stable est trouvé.

4.3 Phase de simulation

Elle doit résoudre les problèmes de *synchronisation* des algorithmes électriques et digitaux, qui ont des gestions différentes du pas de temps.

5. Conclusion :

Ce chapitre a fait en revue une description générale et fondamentale des méthodologies de modélisation et de conception des circuits analogiques et mixtes ainsi que toutes les phases de la simulation analogique-dégitale.

Chapitre II

Simulation analogiques des circuits électroniques

1. Introduction

La simulation électrique est une méthode largement employée dans l'industrie électronique et microélectronique en particulier. En effet, la complexité des calculs manuels caractérisant la prise en compte de divers phénomènes physiques régissant le fonctionnement des dispositifs rend le recours à la simulation électrique très nécessaire.

Nous présentons dans ce chapitre la procédure générale de la simulation analogique des circuits électronique ainsi que divers types d'algorithmes de modélisation des systèmes composés des circuits analogiques.

2. Notions sur la simulation analogique

- Un simulateur électrique est un programme informatique, qui à partir de la description d'un circuit et de ses variables d'excitation permet de calculer n'importe qu'elle caractéristique ou variable électrique (tension, courant, impédance,…) en n'importe quel endroit d'un circuit et quelles que soient les excitations appliquées.

- Le logiciel SPICE est la référence en matière de simulateur analogique des circuits intégrés, développé à Berkeley [1]. Il a donné lieu à de nombreuses versions industrielles basées sur le même langage de description structurelle comme PSPICE, LTSPICE, ORCAD-SPICE,...etc. Une bibliothèque de composants qui sont modélisés dans le code même du logiciel de simulation est fournie comportant des éléments passifs (résistances, capacités, inductances, inductances mutuelles), des composants semi-conducteurs (diodes, transistors bipolaires, à effet de champ JFET et MOSFET), des sources idéales indépendantes de tension et de courant et enfin des sources idéales contrôlées.
- Grâce à un logiciel de simulation électrique, on peut vérifier la conformité des résultats expérimentaux sans passer par la phase d'élaboration du prototype, surtout pour les circuits de grande complexité comportant un grand nombre d'éléments non linéaires.

3. Procédure interne d'un simulateur électrique

3.1 Principe de base d'un simulateur

En général, les langages de simulation analogique se servent de modèles d'équations représentant le comportement physique de différents composants. La précision de ces modèles dépend de la complexité des équations et aussi des nombre de paramètres.

Les simulateurs analogiques sont consacrés à analyser le contenu fréquentiel et temporel des circuits. De ce fait, ils disposent des algorithmes de résolution numériques des équations différentielles.

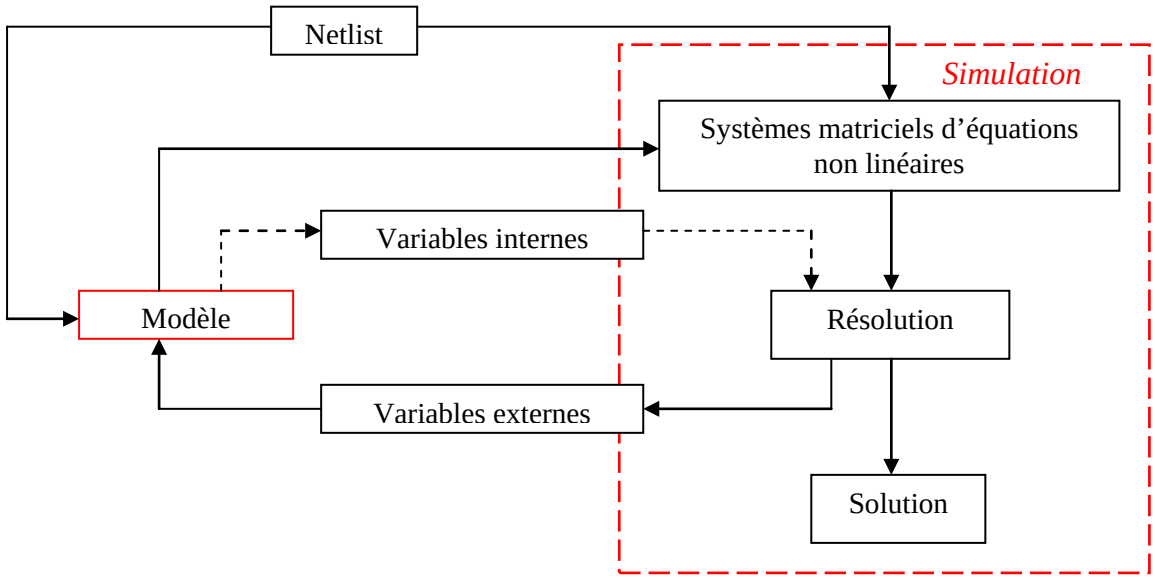


***Figure II.1**: Principe général d'un simulateur électrique.*

Figure II.1 montre que les circuits sont décrit par une liste des interconnexions, appelée *Netlist*, indiquant comment sont connectés le composants. A chaque modèle, un système d'équations décrivant les lois aux différents nœuds (tension et courant) est associé. Le simulateur résout ces systèmes d'équations non linéaires par des méthodes d'intégration numérique, des techniques itératives et des méthodes de résolution matricielles.

3.2 La mise en équation des circuits électriques

Un circuit électrique constitué d'un ensemble de branches, peut être décrit par un système d'équations satisfaisant aux équations de Kirchhoff des courants et des tensions, ainsi qu'aux équations des composants. Chaque composant, considéré comme une interconnexion de branches, exprime les relations des tensions ou courants de ses branches, en fonction des inconnues du circuit.

A partir de la figure II.2, le modèle du composant amplificateur composé par deux branches *AB* et *CD* et par conséquent se caractérise par deux relations définissant les courants qui y circulent.

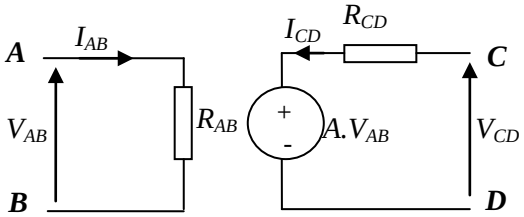


Figure II.2*: Modèle d'un amplificateur linéaire.*

Pour exprimer le système d'équations correspondant à ce réseau électrique, l'algorithme de l'*analyse nodale* est souvent utilisé. En effet, cette méthode conduit à un système d'équations très simple, dont les variables indépendantes du système sont les tensions des nœuds, calculées par rapport à un nœud de référence (la masse). Dans ce cas, le système d'équations est composé des équations de Kirchhoff des courants en chaque nœud, à l'exception du nœud de référence. Le système d'équations est donc exprimé comme suit :

$$\begin{cases} I_{AB} = \frac{V_{AB}}{R_{AB}} \\ I_{CD} = \frac{V_{CD} - A.V_{AB}}{R_{CD}} \end{cases} \quad (1)$$

Cette méthode présente cependant une limitation dans le cas d'utilisation de sources de tension dont le courant qui les traverse est une inconnue. Pour remédier ce problème, l'utilisation des méthodes hybrides, telles que l'*analyse nodale modifiée* (Modified Nodal Analysis ou MNA) s'est avéré nécessaire: cette méthode, utilisée dans SPICE, offre un bon compromis simplicité/généralité. Les variables indépendantes du système sont constituées des tensions des nœuds et des courants circulant dans les sources de tension et dans tous les composants dont les équations de branche ne peuvent être représentées sous la forme explicite :

$$i = Y.v \quad (2)$$

où *Y :* représente l'admittance de la branche.

Les équations du système sont donc constituées des équations de Kirchhoff des courants et des équations de branches pour lesquelles le courant est une inconnue. Enfin, quelle que soit la méthode utilisée, la taille du système à résoudre dépend directement du nombre de nœuds du réseau électrique.

4. Algorithmes de modélisation des circuits analogiques

Dans un circuit analogique, on trouve différents types d'éléments linéaires (résistances), non linéaires (diodes, transistors,…) et/ou des éléments à mémoire (capacités, bobines,…). Du fait des comportements différents de ces éléments, plusieurs types d'analyse peuvent alors

être réalisés par le simulateur. Dans chaque type d'analyse, la résolution des systèmes d'équations est effectuée selon des algorithmes adaptés.

4.1 Etude du point de fonctionnement et analyse DC

Ces deux types d'analyse consistent à déterminer la valeur des tensions et des courants du circuit électrique en régime permanent (indépendamment du temps) et en continu (DC). Le système d'équations extrait est souvent non-linéaire et doit être résolu par des algorithmes itératifs de type *Newton-Raphson* ou de *relaxation* [2].

a. Algorithme de Newton-Raphson

Dans les programmes de type SPICE, l'algorithme de *Newton-Raphson* représente une méthode itérative adaptée aux circuits analogiques fortement couplés. Cet algorithme est souvent utilisé pour la résolution des équations non-linéaires à une inconnue (*f(x)=0)* et il est couramment nommé *la méthode de la tangente*. Le principe de cet algorithme consiste à choisir pour une variable inconnue à l'itération p+1 (x^{p+1}) la racine ξ de l'application affine tangente en x^p à la fonction f :

$$f(x^p) + f'(x^p).(\xi - x^p) = 0 \qquad (3)$$

Cette méthode peut être étalée à la résolution d'un système d'équations non-linéaires en remplaçant l'équation scalaire précédente par un système linéaire où x est le vecteur inconnu et désigne le Jacobien de f . Cependant, La fonction f' calculée dans le cas des circuits électriques comporte un grand nombre de termes nuls. Par conséquent, au lieu de recourir au Jacobien, chaque relation de branche du circuit est linéarisée une par une pour construire un nouveau réseau basé sur des schémas équivalents des éléments linéarisés.

La figure II.3 illustre l'exemple du schéma équivalent linéarisé d'une diode (composant fortement non-linéaire). Ce schéma est constitué d'une conductance g_{eq} et d'une source de courant i_{eq} dont ses valeurs à une itération donnée dépendent des résultats de l'itération précédente.

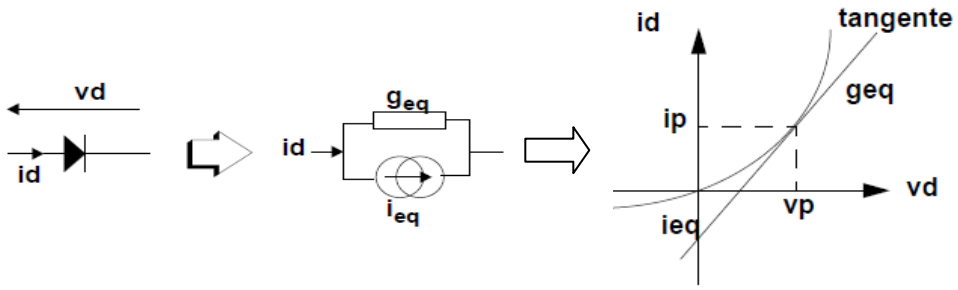


***Figure II.3**: Schéma équivalent linéarisé d'une diode.*

Dans ce cas, le courant traversant la diode doit être modélisé comme suit :

$$i_d = i_{eq} + g_{eq}.v_d \tag{4}$$

Avec :

$$\begin{cases} g_{eq} = \frac{\partial i_d}{\partial v_d}\Big|_{v^p} \\ i_{eq} = i^p - g_{eq}.v^p \end{cases} \tag{5}$$

Le nouveau système linéarisé est ensuite résolu par une simple élimination de *Gauss* ou par factorisation triangulaire *L.U.* (*Lower/Upper*).

L'inconvénient major de cet algorithme est lié à la nécessité de la définition analytique des dérivées partielles de chaque relation de branche. Cette contrainte représente un obstacle à l'écriture éventuelle de modèles comportementaux.

b. Relaxation

Dans le cas des circuits peu couplés (circuits composés de transistors MOS), les algorithmes de *relaxation*, tels que celui de *Gauss-Seidel* [2], donnent de bons résultats de modélisation. Ils consistent en un découplage des variables qui sont calculées une par une en fixant les autres à leurs valeurs précédentes.

Par exemple, l'algorithme de *Gauss-Seidel* calcule les composantes du vecteur inconnu x à l'itération p+1 (x^{p+1}), dans l'ordre des indices croissants: chaque x_i^{p+1} est solution de l'équation scalaire $f_i\ (x_1^{p+1}, x_2^{p+1},\ldots, \xi, x^p_{i+1},\ldots, x_n^p) = 0$, où f_i est la $i^{ème}$ composante de la matrice F du système et ξ est une inconnue. On remarque que cette équation met en jeu les composantes calculées à l'itération courante $x_1^{p+1}, x_2^{p+1}, \ldots, x_{i-1}^{p+1}$ et ceux calculées à l'itération précédente $x^p_{i+1}, x^p_{i+2},\ldots, x_n^p$. Enfin, cette équation scalaire peut être résolue soit:

• par un pas de la méthode de *Newton* (algorithme de *Gauss-Seidel-Newton*):

$$x_i^{p+1} = x_i^p - \frac{f_i(x_1^{p+1}, x_2^{P+1}, \ldots, x_i^p, x_{i+1}^p, \ldots, x_n^p)}{\frac{\partial}{\partial x_i} f_i(x_1^{p+1}, x_2^{P+1}, \ldots, x_i^p, x_{i+1}^p, \ldots, x_n^p)} \quad (6)$$

• par la méthode du *point fixe* ou la méthode de la *sécante*. Au contraire de *Newton*, ces deux algorithmes ne nécessitent pas de dérivées partielles ce qui est intéressant pour la modélisation comportementale.

4.2 Analyse temporelle

L'analyse temporelle consiste en la résolution d'un système d'équations différentielles ordinaires et non-linéaires en un certain nombre de pas de temps. Dans ce cas, on distingue aussi des méthodes itératives et des méthodes de relaxation selon le niveau de découplage des variables.

a. Méthodes itératives

La procédure itérative d'une simulation temporelle SPICE est décrite par la figure II.4. Dans ce cas, un système d'équations algébriques non-linéaires (*discrétisées)* est élaboré à chaque pas de temps, en fonction des valeurs précédemment calculées en utilisant des *algorithmes d'intégration.* La largeur des pas de temps est déterminée dynamiquement par le simulateur en fonction de certains critères de précision:

- Lorsque l'erreur de la solution à un pas de temps donné est inférieure à une certaine valeur, le simulateur tente d'augmenter le prochain pas de temps Δt, afin de réduire la durée de la simulation.

- Lorsque les critères de convergence ne sont pas vérifiés, le pas de temps qui avait été initialement choisi est réduit et une nouvelle itération est effectuée.

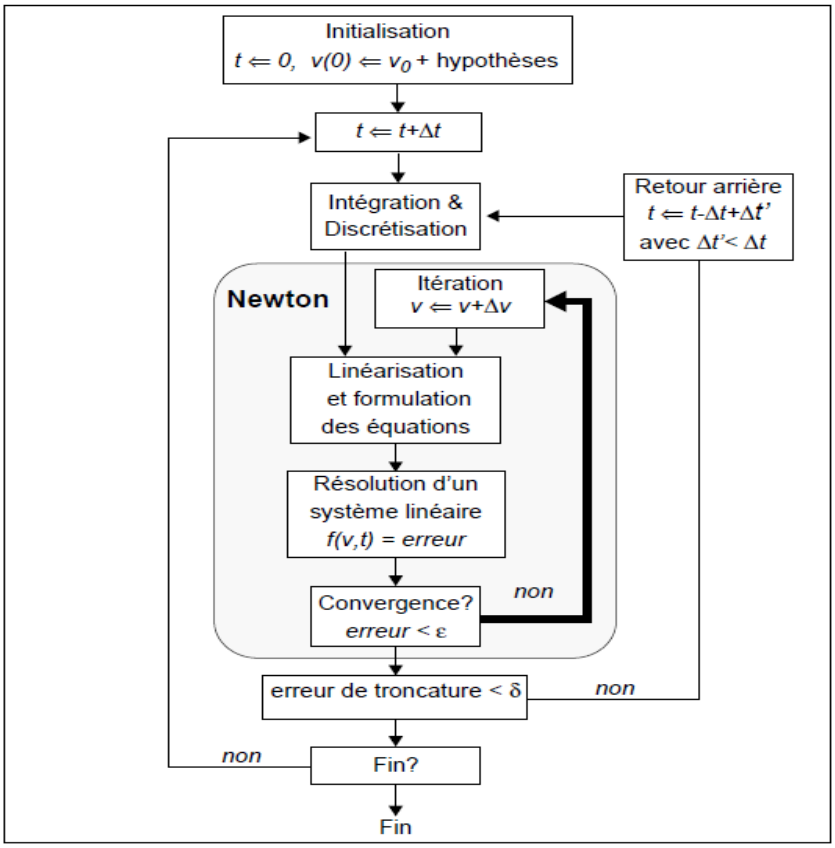


***Figure II.4**: Schéma d'une simulation temporelle SPICE [1] [3].*

Les algorithmes d'intégration se classent en deux catégories: intégration explicite, nommée *Euler-directe* (*forward-Euler*) qui est particulièrement instable, ou implicite (*backward-Euler*, *Gear*, intégration *trapézoïdale*). Ces algorithmes sont caractérisés par leur précision et stabilité.

Le schéma de l'algorithme d'intégration de type *backward-Euler* s'exprime par les développements de Taylor de x_{n+1} (valeur de x à l'instant t_{n+1}) et de sa dérivée $\frac{dx_{n+1}}{dt}$ en fonction des grandeurs calculées à l'instant précédent t_n (x_n et$\frac{dx_n}{dt}$) :

$$x_{n+1} = x_n + h_n . \frac{dx_{n+1}}{dt} - \frac{h_n^2}{2} . \frac{d^2 x_n}{dt^2}(\xi) \qquad (7)$$

où h_n correspond au pas de temps du simulateur entre t_n et t_{n+1} et le terme du second ordre constitue l'*erreur de troncature* qui caractérise la précision de la méthode.

Par exemple, l'équation du courant traversant une capacité ($i = C . \frac{dV}{dt}$) est transformée selon ce schéma à l'équation discrétisée à l'instant t_{n+1} suivante :

$$i_{n+1} = \frac{C}{h_n} . (V_{n+1} - V_n) \qquad (8)$$

b. Méthodes de relaxation

En cas des méthodes de relaxation, le découplage des variables peut être effectué soit au niveau du système d'équations non-linéaires soit au niveau supérieur du système d'équations différentielles.

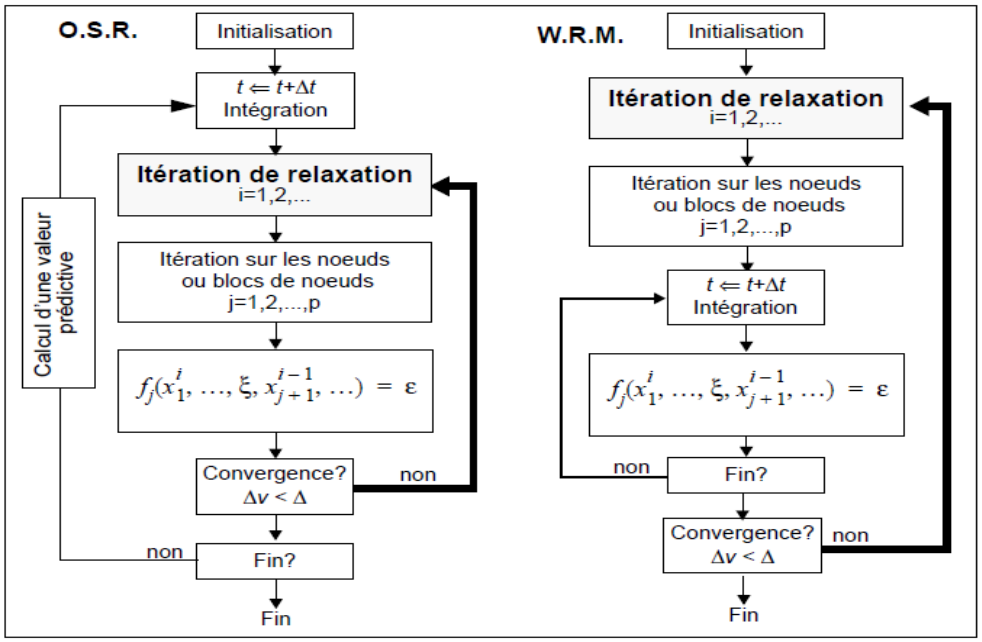


Figure II.5 : *Algorithmes de relaxation: O.S.R. et W.R.M.*

Le premier cas correspond à l'algorithme de relaxation *O.S.R.* de Eldo présenté en Figure II.5 [4]. L'ordre d'imbrication des boucles est: *temps*, *relaxation*, *noeuds*. Une valeur prédictive est calculée à la fin de chaque pas de temps comme condition initiale pour le suivant.

Le second cas constitue l'algorithme *W.R.M.* ou *Waveform Relaxation Method*, utilisée par le simulateur *Relax* [4] (Figure II.5). Ici, l'ordre d'imbrication des boucles est: *relaxation*, *noeuds*, *temps*. Chaque variable est une fonction du temps $x_j(t)$, définie à chaque pas de temps de la simulation, et qui est déterminée à partir des autres variables elles-aussi fonctions du temps. Par rapport au schéma précédent, la boucle de relaxation est simplement déplacée à un niveau supérieur. Cependant, ce type de méthode requiert la mémorisation de toutes les fonctions $x_j(t)$ et en pratique, la simulation est décomposée en plusieurs fenêtres temporelles successives.

4.3 Analyse fréquentielle ou AC

L'analyse AC effectuée par un simulateur de type SPICE correspond à une *étude linéaire* de la réponse fréquentielle du circuit pour des sources sinusoïdales de même fréquence, mais de phases éventuellement différentes. Chaque élément non-linéaire est tout

d'abord remplacé par un modèle équivalent *linéarisé*, dont les valeurs sont déterminées par une analyse initiale du point de fonctionnement. Le système d'équations différentielles est alors transformé en un système d'équations algébriques appartenant au domaine complexe par transformation de Fourier et en effectuant les substitutions: $\frac{dv}{dt} \rightarrow j\omega V$, où ω est la pulsation des stimuli en rad/s. Les termes réactifs, dépendant de ω, sont ensuite évalués aux diverses fréquences de l'analyse.

5. Conclusion

Ce chapitre a été destiné à la simulation analogique des circuits électroniques. Les différentes méthodes de modélisation d'un simulateur électrique de type 'SPICE' ont été expliquées dans le cas de dives modes d'analyse des circuits électriques.

Chapitre III

Les langages de description des systèmes mixtes

1. Introduction

La simulation devient une tâche fondamentale dans le domaine de conception des circuits intégrés VLSI en tant qu'outil de validation des choix du concepteur. En plus, elle doit être la plus rapide et la plus fiable possible pour des raisons de compétitivité. On distingue généralement trois types de simulation:

- *la simulation électrique (analogique)*: ce type de simulation traite des signaux continus dans le temps et est utilisée pour déterminer les performances électriques des circuits. Le niveau de précision demandé est généralement élevé et de ce fait, le temps de simulation des circuits comportant un grand nombre de transistors est souvent critique. Les modèles comportementaux analogiques peuvent être représentés par un ensemble simplifié d'équations différentielles, des fonctions mathématiques non-linéaires ou linéaires par morceaux ou des tables de données.
- *la simulation numérique (digitale):* elle est couramment utilisée pour les circuits digitaux VLSI. Les signaux manipulés ne sont plus électriques mais abstraits: des bits définis par des états logiques (0/1/ indéterminé) ou des mots de bits ou même des fichiers de données peuvent être transmis entre modèles.
- *la simulation mixte (analogique-digitale):* elle permet d'étudier le comportement temporel de systèmes complexes en un temps extrêmement réduit par rapport à une simulation uniquement électrique. Ce type de simulation est en effet basé sur l'abstraction de la partie digitale à un niveau fonctionnel logique. Pour cette partie, les grandeurs étudiées ne sont donc plus électriques mais numériques et sont caractérisées par leurs changements d'état. Des algorithmes dirigés par évènements (*event-driven)* permettent d'étudier de manière très efficace l'évolution des signaux digitaux.

2. Simulation des signaux analogiques-numériques

Le logiciel PSPICE représente un simulateur complet pour la conception analogique. Du fait de ses modèles internes et ses bibliothèques largement rependues et développées, tous les systèmes de haute fréquence jusqu'aux circuits intégrés de basse puissance peuvent être simulés. Dans sa bibliothèque, des modèles peuvent être édités et/ou des modèles de nouveaux dispositifs peuvent être créés à partir des fiches techniques.

La version plus élaborée de PSPICE dite « PSPICE A/D Basics » est un simulateur de signaux mixtes pouvant être employée pour simuler des systèmes contenant des parties

analogiques et des éléments numériques sans limite théorique de la taille. Cependant, quand il s'agit de grands systèmes, les simulations deviennent trop lourdes et demandent un temps d'exécution exagéré.

Par conséquent, les exigences de la technologie et du marché ont imposé le développement d'outils plus vigoureux capables de traiter simultanément les domaines analogiques et numériques. Ce besoin a entraîné depuis la fin des années 90, l'apparition de langages de description matérielle de systèmes à signaux mixtes MSHDLs (*Mixed Signals Hardware Description Langages*) [5]. Ces types de langages offrent un grand intérêt dans une approche de conception système.

3. Langages de modélisation numériques

Les langages de modélisation numérique sont les langages de description matérielle de haut niveau communément appelé HDL (*Hardware Description Language*). Ce dernier représente une instance d'une classe de langage informatique ayant pour objectif la description formelle d'un système électronique. Le HDL est un langage de description de circuits logiques en électronique, utilisé pour la conception des circuits ASICs (*Application Specific Integrated Circuits*) et FPGAs (*Fields Programmable Gate Array*s) [6].

Parmi les fonctions pouvant être réalisées par le langage HDL, on peut citer :

- décrire le fonctionnement et la structure du circuit,
- assurer sa documentation,
- permettre des preuves formelles,
- aider à co-vérifier de netlist ,
- tester le circuit en le vérifiant par simulation.

Il existe plusieurs types de langages de description de matériel (HDL), dont les plus connus sont :

3.1 Langage Verilog

Le langage Verilog était à l'origine un langage propriétaire (non libre) de description de matériel, développé par la société "Cadence Design Systems", pour être utilisé dans leurs simulateurs logiques. L'Institut d'Electrique et d'Electroniques des Ingénieurs (IEEE) a normalisé le langage « Verilog » comme un langage de description pour les modèles numériques, cette norme est définie par le standard : IEEE Std 1364-2005 [7].

Le langage Verilog permet d'une part de décrire l'enchainement d'événements et d'autre part de synthétiser des circuits numériques par combinaison de plusieurs éléments

logiques (modules, portes logiques,…). La syntaxe de « Verilog » est largement inspirée du langage de programmation C [8].

3.2 Langage VHDL

Le langage VHDL (*Very High Speed integrated circuits Hardware Description Langage*) est un langage lisible, actuel et puissant de description de matériel destiné à décrire le comportement et/ou l'architecture d'un système électronique numérique.

L'intérêt d'une telle description réside dans son caractère exécutable, c.-à-d. une spécification fonctionnelle décrite en VHDL peut être vérifiée par simulation avant la finalisation de la conception détaillée du système. La syntaxe du VHDL est originaire du langage ADA.

Exemple: Décodeur 2 vers 4

Nous considérons un circuit combinatoire qui est un décodeur 2/4 (2 entrées et 4 sorties) représenté par la figure III.1. Il est réalisé avec des portes « inverseur » et des portes « AND ». Dans ce cas la modélisation numérique par le langage VHDL est illustrée sur l'encadré III.1.

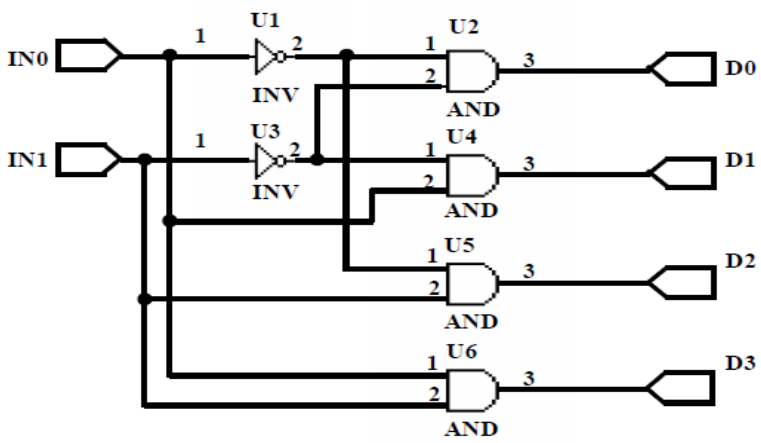


Figure III.1: *Décodeur 2/4.*

IN0	IN1	D0	D1	D2	D3
0	0	1	0	0	0
0	1	0	0	1	0
1	0	0	1	0	0
1	1	0	0	0	1

Tableau III.1: *Table de vérité du décodeur 2/4.*

```
entity décodeur is
--Définition des entrés sorties
…
end entity décodeur ;
architecture description of décodeur is
begin
IN0 <= not(IN0) after1.0 ms;
IN1 <= not(IN1) after2.0 ms;
D0 <= not(IN0) and not(IN1);
D1 <= IN0 and not(IN1);
D2 <= not(IN0) and IN1;
D3 <= IN0 and IN1;
end description;
```

Encadré III.1: *Code VHDL du décodeur 2/4.*

4. Langages de modélisation mixte multi-domaine

Un modèle mixte inclut des parties ayant un comportement continu (parties analogiques) et des parties ayant un comportement dirigé par événements (parties logiques) qui interfèrent. À l'aide des langages de modélisation mixte de haut niveau, les différentes phases de conception d'un système peuvent être optimisées. Ces langages permettent de traiter indifféremment des modélisations logiques, analogiques ou mixtes au sein d'un même composant ou système.

Parmi ces langages, ont peut citer : VHDL-AMS, Verilog-AMS, MAST, Modelica et la notation Bond Graph.

4.1 MAST

Ce langage est propriétaire qui a été proposé en 1986 par la société *Analogy* (*Synupsys* aujourd'hui). MAST représente la première tentative réussie de définition d'un langage de description comportementale réellement orienté systèmes multi-technologiques pour la simulation analogique et la simulation des signaux mixtes offrant la possibilité de modéliser des systèmes électriques et non-électriques (thermiques, hydrauliques, optiques…). Ce langage est beaucoup employé dans le domaine de l'industrie, notamment dans de nombreuses entreprises automobiles et aéronautiques. Cependant, ses principaux inconvénients sont : son absence de normalisation IEEE, sa syntaxe est éloignée de VHDL et Verilog, sa mauvaise adaptation à la description des systèmes numériques et aussi ses fonctions cachées.

4.2 Verilog AMS

Verilog-AMS est un langage de description du matériel (HDL) permettant d'exprimer de modèles et de systèmes à temps continu et à temps discret. Il a été créé sous la tutelle d'Accellera (Organisation de standards EDA : *Electronic Design Automation*) afin de mettre en place les extensions analogiques mixtes de Verilog (IEEE-1364) [9]. La version la pus récente est « Verilog-AMS LRM-2.2 » sortie en novembre 2004.

Le langage Verilog-AMS permet de faire la description comportementale des systèmes analogiques et mixtes. Ainsi, ce langage peut être applicable aux systèmes électriques et non électriques. Des descriptions de systèmes peuvent être effectuées sous ce langage, en utilisant des concepts comme des nœuds, des branches et des ports. Les signaux de type analogique et numérique peuvent être présents dans le même module. Cependant, le Verilog-AMS n'est pas une norme IEEE.

4.3 VHDL-AMS

La norme VHDL a dû évoluer pour répondre aux différents besoins de l'électronique. L'IEEE–DASC (IEEE-Design Automation Standards Committee) a créé la norme IEEE 1076.1-1999, plus connue sous le nom VHDL-AMS (VHDL-Analog & Mixed Signal). Cette nouvelle norme est une extension de la norme IEEE 1076-1992 déjà existante. Elle a été complétée et définie actuellement par le standard : IEEE Std 1076.1 2007, qui a été publié le 15 novembre 2007 [10].

Le langage VHDL-AMS inclut toutes les propriétés du VHDL standard, avec en plus, l'aptitude de décrire les systèmes mixtes à travers de modèles multi abstractions, multidisciplinaires et/ou hiérarchiques à temps continu et à événements discrets [11] (Figure III.2).

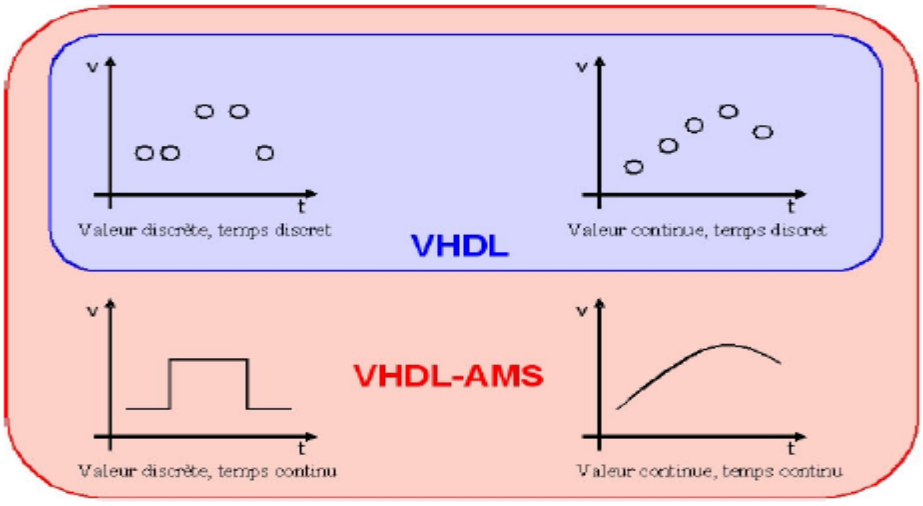


Figure III.2 : Couverture du langage VHDL-AMS.

VHDL-AMS permet de modéliser tout système dont le comportement peut être décrit par un ensemble d'équations différentielles algébriques et/ou ordinaires (ODE, ADE) qui ont le temps comme variable indépendante. Ces équations peuvent être écrites sous la forme : $F(x, dx/dt,\ldots, t)=0$, où x est le vecteur d'inconnues.

L'avantage de ce langage non propriétaire est de proposer un langage commun indépendant des fournisseurs et de la technologie. Du point de vue technique, il permet une haute modularité facilitant des descriptions hiérarchiques. Tous les systèmes qui peuvent être modélisés sous Matlab, VHDL et Spice peuvent aujourd'hui être modélisés sous un seul langage (VHDL-AMS). En effet, ce dernier peut travailler simultanément comme un simulateur à évènements discrets et un solveur d'équations différentielles.

4.4 Modelica

Modelica représente un langage de modelisation informatique orienté objet, consacré à la modélisation de systèmes physiques, complexes et hétérogènes. Les modèles Modelica sont décrits mathématiquement de facon acausale par les équations différentielles algèbriques discrètes [12]. Ainsi, l'effort de modélisation est considérablement diminué avec ce langage du fait de la réutilisation possible des composants et aussi de l'inexistence des manipulations manuelles. L'utilisation du langage Modelica est presque similaire à celle de Verilog-AMS et VHDL-AMS car il décrit un système sous la forme d'un ensemble d'équations.

5. Conclusion

Une présentation des langages de programmation des systèmes contenant des éléments purement numérique a été effectuée au début de cette partie. Nous avons également exposé des langages de description de haut niveau ayant l'aptitude de décrire les systèmes mixtes à travers de modèles multi abstractions, multidisciplinaires à temps continu et à événements discrets.

Chapitre IV

Conception des systèmes microélectroniques analogiques- numériques via VHDL-AMS

1. Introduction

L'objectif du standard VHDL-AMS est de fournir un outil de description hiérarchique de simulation des systèmes continus et mixtes (analogique/numérique) en conservation d'énergie ou non. Le langage devait supporter la modélisation à différents niveaux d'abstraction en domaine électrique et non électrique (systèmes constitués d'éléments hydrauliques, thermiques… etc.). Les circuits à modéliser sont descriptibles par des systèmes d'équations différentielles et algébriques (EDA). La résolution de ces systèmes devait inclure la gestion des discontinuités. D'autre part il fallait respecter les exigences au niveau des interactions entre partie numérique et partie continue des systèmes mixtes. Il apparu donc que la spécificité des comportements analogiques et des systèmes mixtes devrait entraîner la création d'un certain nombre de nouveaux éléments:

- création d'un noyau de résolution analogique pour résoudre les systèmes d'équations,
- notation pour les systèmes d'équations,
- création de nouvelles quantités pour exprimer les différences de potentiel aux bornes d'une branche et le courant la traversant ainsi que la notion de tolérance,
- redéfinition du cycle de simulation pour la simulation des systèmes mixtes,
- création des instructions de synchronisation.

2. Environnement de travail du langage VHDL-AMS

La figure IV.1 représente l'environnement de travail du standard VHDL-AMS contenant différentes phases d'édition, d'analyse, d'élaboration et d'exécution :

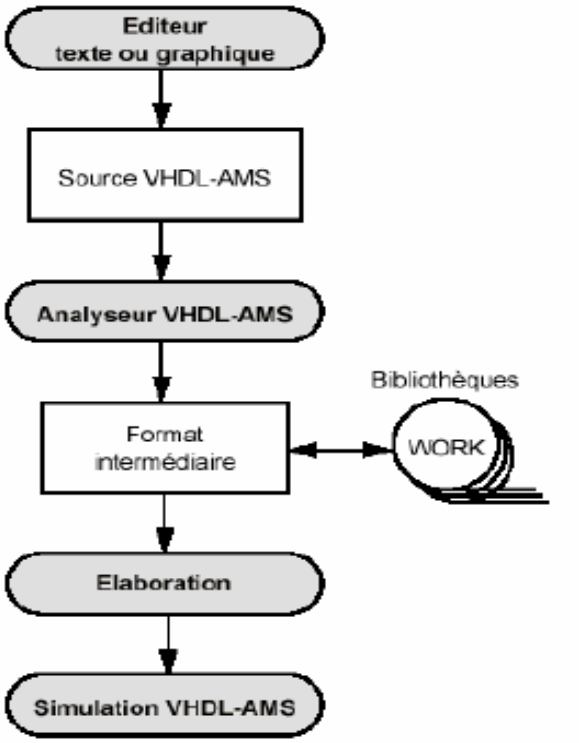


***Figure IV.1*:** *Environnement travail VHDL-AMS.*

2.1 L'interface graphique

Elle 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. 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.

2.2 L'analyseur (compilateur)

Le rôle du compilateur est la vérification de la syntaxe d'une description VHDL-AMS. Il permet la détection d'erreurs locales, qui ne concernent que de l'unité compilée. Plusieurs techniques d'analyse sont actuellement utilisées par les outils du marché.

2.3 Bibliothèque de travail (Working library)

Chaque concepteur possède une bibliothèque de travail de nom logique WORK (le nom est standard) dans laquelle sont placés tous les modèles compilés.

2.4 Le simulateur

Il 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 excitations et les opérations à appliquer aux signaux de sortie pour les visualiser (sous forme texte ou graphique).

2.5 La phase d'élaboration

Elle 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.

3. Description structurelle et configuration d'un modèle VHDL-AMS

3.1 Structure générale d'une entité de conception VHDL-AMS

La modélisation en VHDL-AMS de tout composant s'effectue au moyen de deux types d'objets : entity et architecture constituant tous les deux une entité de conception VHDL-AMS.

a. L'entité (entity)

La déclaration d'entité définit l'interface d'un modèle avec le monde extérieur au moyen des ports. On peut comparer l'entité a une boite noire où seules les entrés-sorties du composant sont visibles. Lors de l'écriture d'une entité, on ne déclare que l'interface avec le monde extérieur grâce aux mots **port** et **generic**.

generic: regroupe la déclaration des paramètres du composant.

port: décrit l'interface avec l'environnement extérieur par l'intermédiaire des nœuds externes.

Exemple de syntaxe :

entity nom_de_l'entité **is**
generic (generic_déclarations);
port (port_déclarations);
end entity nom_de_l'entité

b. L'architecture

Une architecture définit le comportement et/ou la structure du système modélisé. Une architecture se réfère toujours à une entité unique et contient la description de la fonction réalisée par cette dernière. Pour une entité donnée réalisant une fonction précise, il peut y avoir autant d'architectures de manières de décrire la fonction à réaliser (Figure IV.2).

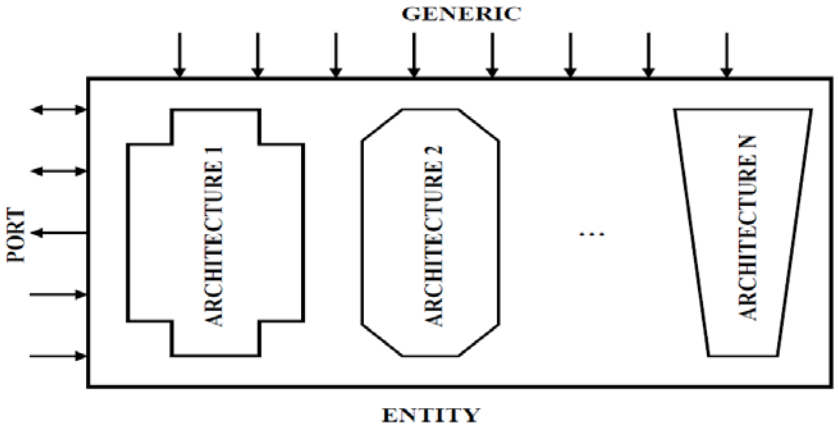


Figure IV.2: *Structure d'un modèle VHDL-AMS.*

Exemple de syntaxe :

architecture nom_de_l'architecture **of** nom_de_l'entité **is**

déclaration_des_variables

begin

déclaration_des_équations

end architecture nom_de_l'architecture

3.2 Configuration générale d'un modèle VHDL-AMS

Une vue interne (architecture) possible en VHDL-AMS est une description structurelle pour laquelle le modèle est une interconnexion de composants, avec éventuellement un nombre de niveaux hiérarchiques non limité.

_ Structure Générale d'un modèle VHDL-AMS _

library : ouverture de bibliothèques

use: utilisation des bibliothèques

entity : spécification d'entité (vue externe du modèle)

is

generic: paramètres génériques

port: ports de connexion

signal (in/out,inout) : Signaux à événements discrets

QUANTITY (IN/OUT) : quantités analogiques à temps continu

TERMINAL : équipotentielle utilisés pour les connexions "Kirchoff"

end entity

architecture: vue interne du modèle

is

signal: déclaration de signaux internes

QUANTITY : déclaration de quantités internes

TEMINAL : déclaration de terminaux internes

begin corps de l'architecture

Instanciation de composants

Instruction concurrente : **Process** signaux

<= Affectation de signal numériques

Assert test et rapport

BREAK synchronisation des simulateurs

INSTRUCTIONS SIMULTANEES = = quantités analogiques

end architecture

a. Quantité (quantity)

Une quantité sert à modéliser une quantité physique électrique, mécanique thermique…etc. Contrairement aux signaux et aux variables qui ne changent de valeur qu'aux instants précis appelés événements les quantités sont des fonctions continues du temps (ou de la fréquence). Les quantités peuvent être :

- Des quantités libres : qui sont déclarées à l'intérieur d'une architecture :

  quantity omega : real ;

- Des quantités sources : Les quantités sources permettent de définir les signaux utilisés pour les analyses en fréquence AC et NOISE.

  quantity iac : **real spectrum** 1.0 , 0.0 ; --Magnitude Phase

  quantity inn : **real noise** 4*k*T/R ;

  i == V/R + iac + inn; -- source de courant de résistance interne R

- Des quantités d'interface : Les quantités d'interface permettent de faire de la modélisation de type schéma-bloc (signal-flow). Les quantités d'interface sont déclarées dans la déclaration de port d'une entité. Les quantités sont de mode in ou out.

  entity ampli **is**

  port(**quantity** Vip, Vin : **in** real);

  quantity Vout : **out** real;

  signal Enable : **in** bit);

  end entity;

- Des quantités de branche : elles sont associées aux terminaux :
  - Les quantités across représentent un effort : différence de potentiel électrique, différence de température, différence de pression ….
  - Les quantités through représentent un flux : courant électrique, puissance themique, débit volumique…

b. Terminaux (terminal)

Un terminal correspond à une équipotentielle d'un système physique conservatif décrit par un graphe. Le domaine physique auquel appartient le terminal est la nature du terminal la déclaration d'une nature définit le type des quantités across et through ainsi que la reference.

Exemple : Les déclarations ci- dessous sont correspondantes au circuit présenté sur la figure IV.3.

```
library disciplines;
use disciplines.electromagnetic_system.all;
terminal T1, T2, T3, T4 : electrical;
quantity V1 across I1 through T1 to T2;
quantity V2 across I2 through T3; -- le deuxième terminal est la
référence
quantity V3 across I3 through T3; -- V3 est identique à V2
quantity V4 across T3 to T4; -- ne crée pas de branche
quantity I4 through T4 ;
```

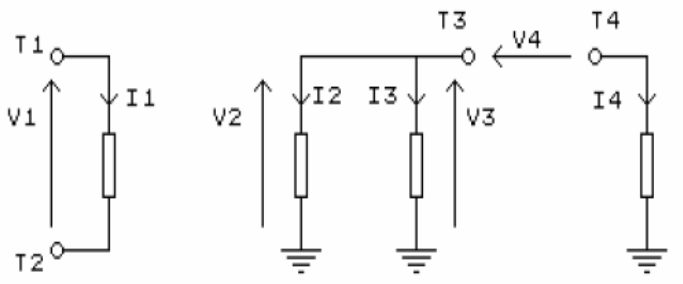


Figure IV.3: *Exemple d'un circuit électrique.*

4. Exemple de modélisation en VHDL-AMS des circuits électriques

4.1 Modélisation des composants élémentaires

- **Résistance :**

Le courant traversant une résistance est défini selon la loi d'Ohm par:

$$i_R(t) = \frac{u_R(t)}{R} \quad (1)$$

Syntaxe :

entity resistor **is**

generic (résistance : real := 1.0); -- déclaration d'une valeur par défaut de la résistance.

port (**terminal** n1, n2 : electrical);

end resistor

architecture behav **of** resistor **is**

quantity r_e across r_i through n1 to n2;

begin

r_i == r_e/résistance; -- Calcul classique du courant à travers une résistance.

end behav;

- **Capacité :**

Le courant traversant un condensateur est décrit par l'expression suivante :

$$i_C(t) = C.\frac{du_C(t)}{dt} \quad (2)$$

Syntaxe :

entity capacitor **is**

generic (capacité : real := 10.0e-9); -- déclarer une valeur par défaut de capacité

port (**terminal** n1, n2 : electrical);

end capacitor;

architecture behav **of** capacitor **is**

quantity c_e across c_i through n1 to n2;

begin

c_i == capacité*c_e'**dot**;

end behav;

- **Inductance :**

Le courant traversant une inductance est exprimé par l'équation :

$$i_L(t) = \frac{1}{L}.\int u_L(t)dt \qquad (3)$$

Syntaxe :

Entity inductor **is**

generic (L : real := 1.0); -- valeur par défaut obligatoire.

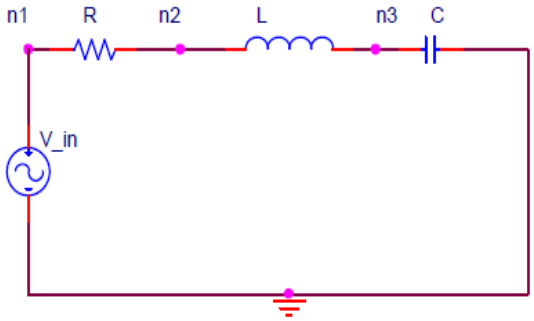
port (**terminal** n1, n2 : electrical);

end inductor;

architecture behav **of** inductor **is**
quantitty L_e across L_i through n1 to n2;
begin
L_i == (L_e'**Integ**)/L; -- utilisation de la quantité implicite Q'Integ
end behav;

4.2 Modélisation d'un circuit RLC en VHDL-AMS

A titre d'exemple, considérons le circuit RLC représenté sur la figure IV.4. La modélisation sous VHDL-AMS de ce circuit consiste tous d'abord à définir chaque composant séparément (source de tension, résistance, inductance et capacité), puis la définition du circuit global comme il est illustrée ci-dessous :



R = 1KW
L = 10 nH
C = 1 nF

Figure IV.4: *Circuit RLC série.*

entity circuit **is**

end;

architecture behavioral **of** circuit **is**

terminal n1,n2,n3: ELECTRICAL;

begin

vsrc: entity voltg (behavioral) port map (n1,

```
electrical_ground);
    r1: entity resistor (behavioral) port map (n1, n2);
    l1: entity inductor (behavioral) port map (n2, n3);
    c1: entity capacitor (behavioral) port map (n3, electrical_ground);
end;
```

5. Modélisation des convertisseurs sous VHDL-AMS

5.1 Convertisseurs Analogique-Numérique

Cahier de Charge : Pour cette application, l'étude de la conversion analogique-numérique se fait par comparaison des quantités avec la tension du seuil. Si une des quantités spécifiées dépasse le seuil d'une amplitude désignée (threshold), avant la fin du calcul de la solution (prochain événement), le calculateur finira le travail prématurément.

Pour n'importe quelle valeur scalaire Q, le booléen *Q' Above(level)* est vrai si Q > level et réciproquement. Un événement attaché à *Q' Above(level)*, intervient à chaque changement de signe de Q- level. Le dépassement d'un seuil peut être utilisé pour la conversion analogique-numérique.

Code VHDL-AMS :

```
entity limiter is
end entity;
architecture beh of limiter is
    constant vmax : real := 1.0;
    constant vmin : real := -1.0;
    quantity vin1, vin2, vout1, vout2 : real;
begin
    if vin1 > vmax use vout1 = = vmax;
    else if vin1 < vmin use vout1 = = vmin;
    else vout1 = = vin1;
    end use;
    vin1 == 3.0*sin(2.0*math_pi*1.0e7*now);
    if vin2'above(vmax) use vout2 == vmax;
    elsif not(vin2'above(vmin)) use vout2 == vmin;
    else vout2 == vin2;
```

```
    end use;
    vin2 == 3.0*cos(2.0*math_pi*1.0e7*now);
end architecture beh;
```

5.2 Convertisseurs Numérique-Analogique

Cahier de Charge : maintenant on va étudier l'application de la conversion numérique-analogique en utilisant les instructions ramp et slew :

- *S'ramp(Tr,Tf) :* Renvoie une quantité qui suit les valeurs du signal S avec un temps de montée Tr et un temps de descente Tf. S doit être de type real. Si Tf est omis, Tf = Tr. Si Tr est omis, il vaut 0 avec Tr et Tf réels (secondes).
- *S' slew(rising_slope, falling_slope):* comportment analogue.

Code VHDL-AMS :

```
entity D2A is
end entity D2A;
architecture beh of D2A is
    constant Vol : real := 0.5;
    constant Voh : real := 4.5;
    quantity Vramp, Vslew : real;
    signal Vin : real := 0.0 ; --Initialisation par défaut à Real_Low
    signal Din : bit :='1';
    begin
        process
        begin
        wait for 100ns;
        Din <= not Din;
    end process;
    Vin <= Voh when Din = '1' else Vol;
    Vramp == Vin'ramp(20.0e-9,10.0e-9);
    Vslew == Vin'slew(0.4e9,-1.0e9);
    break on Vin;
end architecture beh ;
```

6. Conclusion

Dans ce chapitre, nous avons décrit en détail l'utilisation d'un même langage de description matérielle (VHDL-AMS) pour la description des parties analogiques et digitale. Ce dernier a permis une simulation conjointe des différentes parties des systèmes électroniques mixtes, avec une représentation fiable des interactions.

Références Bibliographiques

[1] L.W.Nagel, "*SPICE2: A Computer program to simulate circuits"*, Memorandum No. ERL-M520, Electronics Research Laboratory, University of California, Berkeley, 1975.
[2] R.Raghuram, "*Computer Simulation of Electronic Circuits"*, ISBN 0-470-21331-0 John Wiley & Sons Inc, 1989.
[3] *Spectre Reference Manual* 4.3.4, Cadence Design Systems Inc., June 1995.

[4] B.Hennion, P.Senn, "*Eldo: A new third generation circuit simulator using the One-Step Relaxation method"*, in Proceedings of ISCAS, pp. 1065-1068, 1985.

[5] R. Scott Cooper. "*The Designer's Guide to Analog & Mixed-Signal Modeling*". Avant Corp. ISBN 0-9705953-0-1, 2001.

[6] T. Riesgo, Y. Torrojaand, E. Torre, *" Design Methodologies Based on Hardware Description Languages"*, IEEE Transaction on Power Electronic, Vol. 46, (1), pp. 3-11, February 1999.

[7] IEEE Computer Society, "*IEEE Standard for Verilog Hardware Description Language"*, IEEE Std 1364-2005, New York- USA, pp. 590, 2006.

[8] Homepage of the Verilog modeling language. http://www.verilog.com/

[9] Accellera, Accellera Verilog Analog Mixed-Signal Group, *"Velilog-AMS Home", 1998*. [En ligne]. Adresse URL : http://www.eda.org/verilog-ams/. 1998

[10] IEEE Computer Society, *"IEEE Standard VHDL Analog and Mixed-signal Exten-sions"*, IEEE Std 1076.1-2007, New York-USA, pp. 342, 2007.

[11] Y. Herve, "*VHDL-AMS : Applications et enjeux industriels"*. Dunod-Université - Collection : Sciences-sup - préface d'Alain Vachoux. ISBN : 2-10-005888-6 - mars 2002.

[12] G. Marguerie, "*De Modelica à XMLlab "*, Ingénierie de Systèmes Informatiques Complexes ISICO, EISTI, pp. 41, août 2006.