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 :

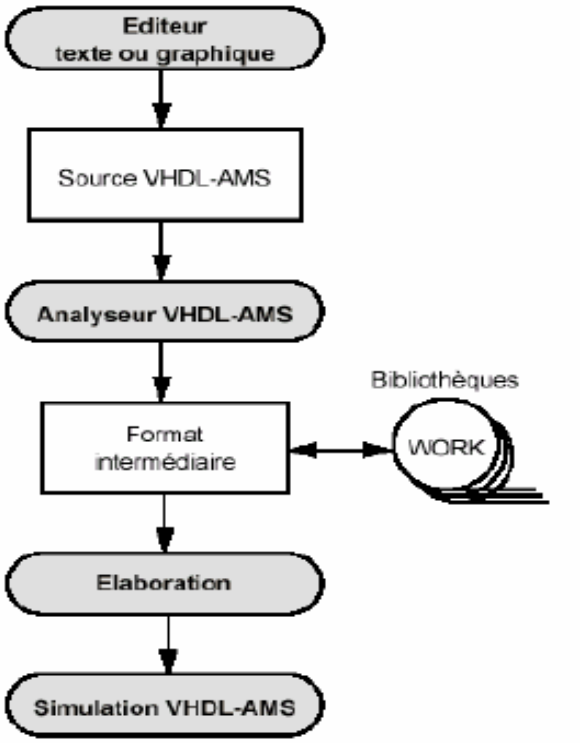


***Figure IV.1*:** *Environnement travail VHDL-AMS.*

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).

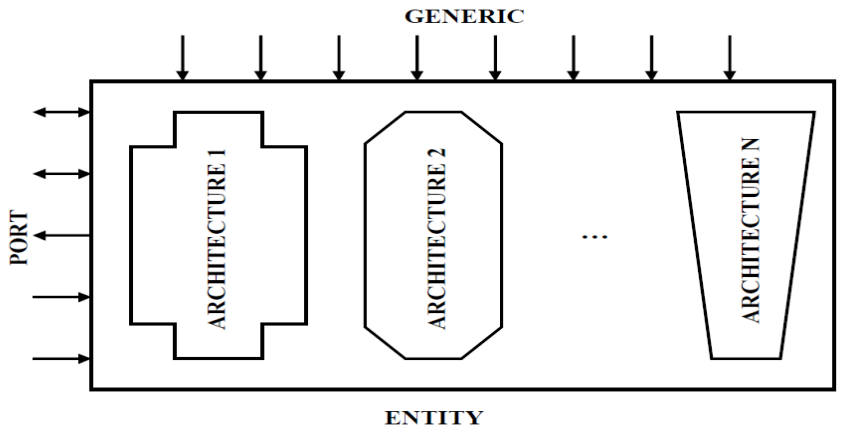


Figure IV.2: *Structure d'un modèle VHDL-AMS.*

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 ;
```

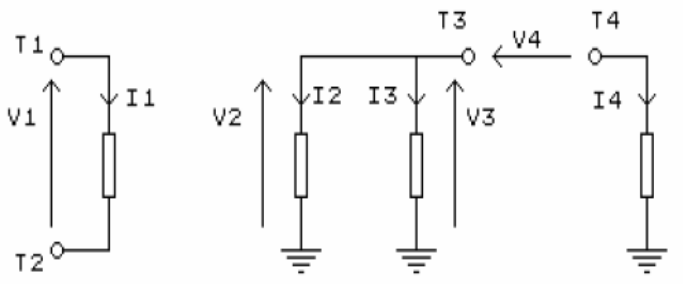


Figure IV.3: *Exemple d'un circuit électrique.*

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.

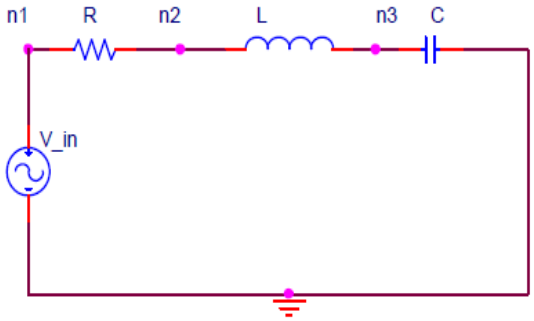
port (**terminal** n1, n2 : electrical);

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.