

## II.1 Introduction :

Dans ce chapitre, nous allons commencer par présenter des notions générales concernant la modélisation et simulation des composants de semi –conducteurs en en utilisant SPICE dans un premier temps, puis dans un deuxième temps le langage de description matérielle : VHDL-AMS. Nous présenterons également quelques résultats de simulation donnés par ces deux outils.

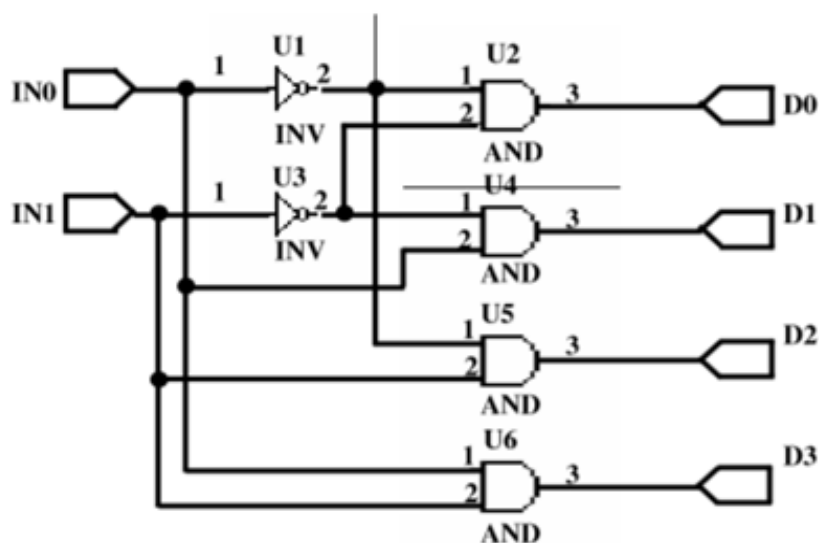
## II.2 Modélisation numérique :

Exemple d'un décodeur 2 vers 4.

Un décodeur est un circuit combinatoire qui a N entrées et  $2^N$  sorties, pour notre exemple on a un décodeur 2 - 4 (2 entrées et 4 sorties). Le décodeur que nous avons présenté est réalisée avec des portes “inverseur” et “AND” (voir la figure II.1), son code sous VHDL est représenté sur yul'encadré II.1.

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 II.1 :** Table de vérité



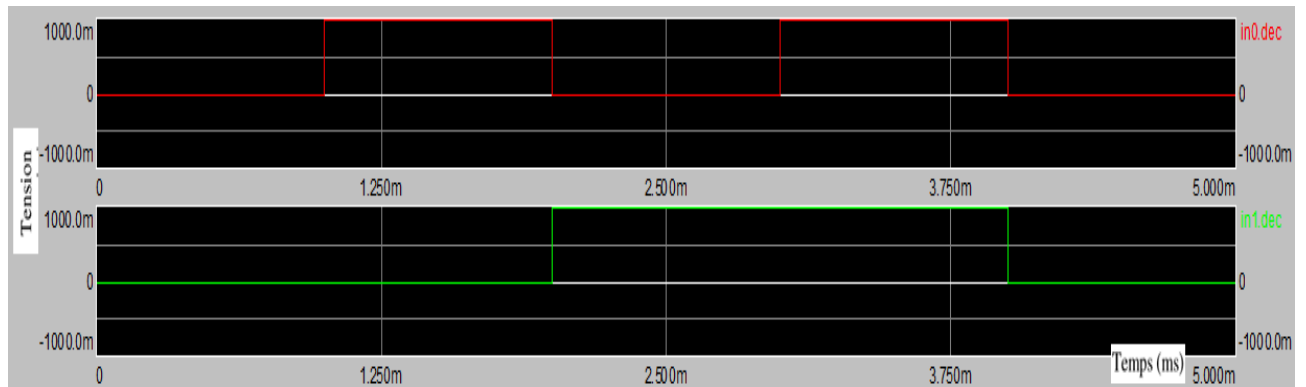
**Figure II.1 :** Décodeur 2/4

```

ENTITY décodeur IS
--Définition de entré sorties ... END ENTITY décodeur;
ARCHITECTURE description OF décodeur IS BEGENTITY décodeur IS
--Définition de entré sorties ... END ENTITY décodeur;
ARCHITECTURE description OF décodeur IS
BEGIN
    IN0 <= not(IN0) after 1.0 ms ;
    IN1 <= not(IN1) after 2.0 ms ;
    D0 <= (not(IN1) and not(IN0));
    D1 <= (not(IN1) and IN0);
    D2 <= (IN1 and not(IN0));
    D3 <= (IN1 and IN0);
END description;

```

Encadré II.1: Code VHDL d'un décodeur



a): Signaux d'entrées.

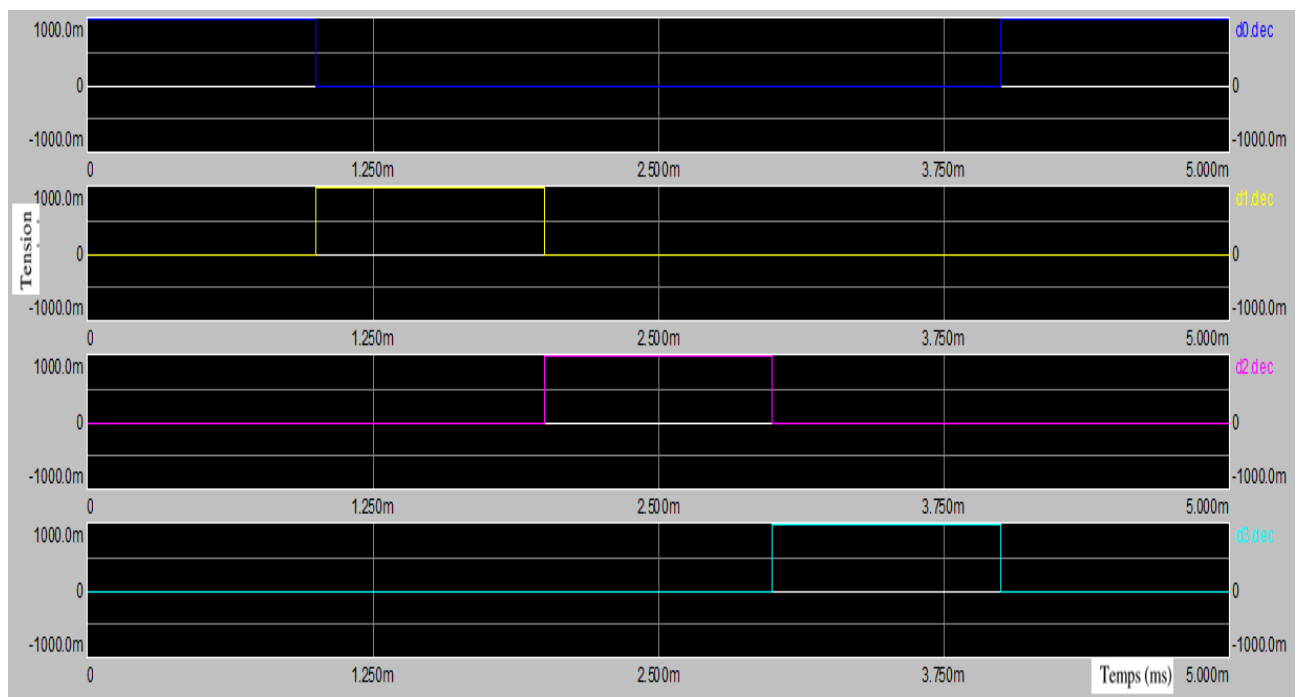


Figure II.2: Résultats de simulation d'un décodeur 2/4. Par VHDL-AMS

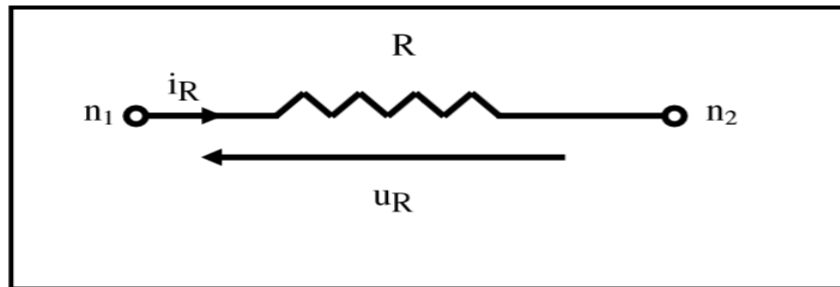
### II.3 Exemple de modélisation de quelques composants élémentaires :

Par différents exemples nous allons montrer les premières applications des nouveaux objets introduits ainsi que leurs utilisations dans leurs contextes [4].

#### a) Résistance discrète :

$$i_R(t) = u_R(t)/R$$

$$u_R(t) = R i_R(t)$$



#### ENTITY resistor IS

GENERIC (résistance: real: = 1.0);

-- On declare l'entité d'une

PORT (TERMINAL n1, n2 : electrical);

simple résistance, on est obligé

END resistor

de déclarer une valeur par

ARCHITECTURE behav OF resistor IS

défaut.

QUANTITY r\_e across r\_i through n1 to n2;

BEGIN

r\_i == r\_e/résistance;

-- Calcul classique du courant à

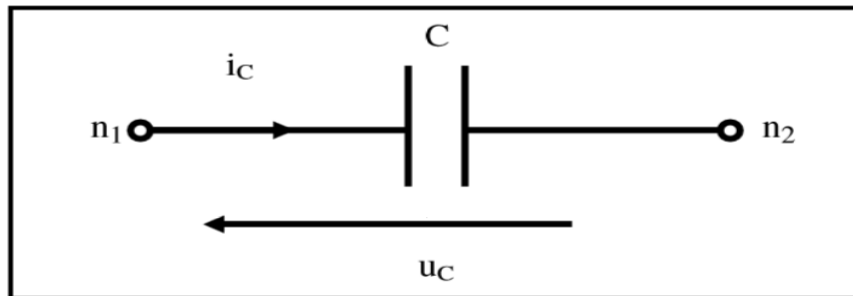
END behav;

**Encadré II.2:** modélisation de résistance

**b) Capacité discrète :**

$$I_C(t) = C \cdot (d/dt) U_C(t)$$

$$U_C(t) = (1/C) \int i_C(t) dt + U_{C0}$$



**ENTITY** capacitor **IS**

**GENERIC** (capacité : real := 10.0e-9); -- On déclare l'entité d'une

**PORT** (**TERMINAL** n1, n2 : electrical); simple capacité, on est

**END** capacitor; obligé de déclarer une

**ARCHITECTURE** behav **OF** capacitor **IS** valeur par défaut.

**QUANTITY** c\_e across c\_i through n1 to n2;

**BEGIN**

C\_i == capacité\*c\_e'Dot; -- Calcul classique du courant

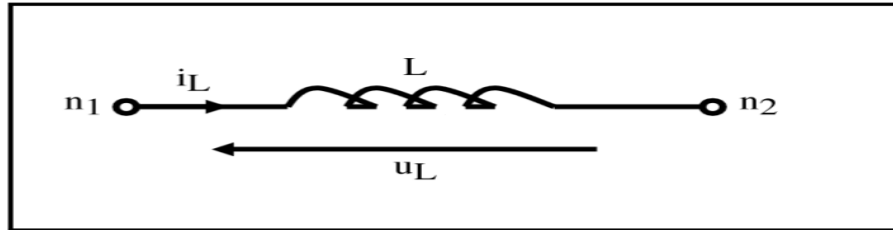
**END** behav; à travers une capacité.

**Encadré II.3:** modélisation de Capacité

**c) Inductance :**

$$i_L = (1/L) \int u_L(t) dt$$

$$u_L(t) = L \cdot (di/dt)$$



**ENTITY** inductor **IS**

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

**PORT** (TERMINAL n1, n2: electrical);

**END**

**ARCHITECTURE** behav **OF** inductor **IS**

**QUANTITY** L\_e across L\_i through n1 to n2;

**BEGIN**

L\_i == (L\_e'Integ)/L;

-- utilisation de la QUANTITY

**END** behav;

implicite Q'Integ.

**Encadré II.4:** modélisation de Inductance

**II.3.1 l'instruction (Test bench du circuit) :**

Le processus d'analyse permet d'extraire ou de vérifier les propriétés d'un système. La simulation est un type d'analyse dynamique pour lequel le modèle est soumis à un ensemble de stimulais. Ceci génère un certain nombre de réponses qui permettent d'extraire des propriétés du modèle est donc du système simulé. Un modèle exécutable définit une procédure de calcul qui a pour effet de (simuler) un ensemble de propriétés d'un système. C'est un simulateur, usuellement réalisé sous la forme d'un programme logiciel, qui calcule des réponses à des stimulais appliqués de l'extérieur du modèle. Un modèle de test (test bench) encapsule le générateur de stimulais, le modèle à tester et le traitement des réponses [7].

### II.3.2 Instructions simultanées :

Les instructions simultanées sont utilisées pour indiquer des équations différentielles et algébriques. Elles sont permises dans tout le corps des instructions d'un bloc.

#### a) l'instruction simultanée simple :

Nous avons déjà rencontré de telles instructions dans les exemples du paragraphe précédent

«Exemple de syntaxe». Ces instructions sont représentées par

le symbole : '='.

Syntaxe :

Expression == expression;

#### b) l'instruction simultanée IF :

Exemple de syntaxe :

**IF** condition **USE**

Partie simultanée

**ELSIF** condition **USE**

-- Plusieurs structures **ELSIF** partie

simultanée peuvent s'enchaîner sans

**ELSE** aucune restriction.

Partie simultanée

**END USE**

Les expressions explicites de l'instruction **IF** sont celles de la partie simultanée qui doit être effectuée selon les résultats des conditions [4].

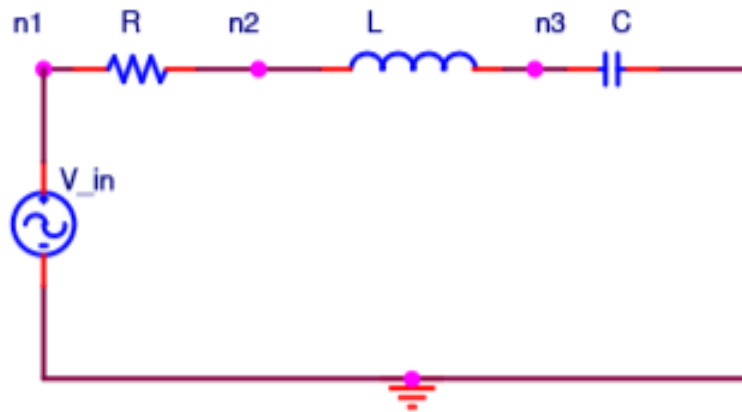
### II.4 Résultat de simulation d'un circuit RLC :

Il existe plusieurs types de circuits RL, RC, RLC. A titre d'exemple nous présentons le résultat de simulation d'un circuit RLC série (figure II.3), les résultats de modélisation sous VHDL-AMS sont présentés sur la figure II.4.(a), et sous PSPICE sont présentés sur la figure II.4.(b). Les valeurs numériques sont les mêmes pour les deux applications:

$$R = 1\text{K}\Omega$$

$$L = 10\text{ nH}$$

$$C = 1\text{ nF}$$



**Figure II.3:** Circuit RLC série.

Le circuit RLC considéré est défini sous SPICE par la net liste de l'encadré II.5, et sous VHDL AMS avec le code représenté sur l'encadré II.6.

**Encadré II.5:** La net liste de SPICE qui représente le circuit de la figure II.3

```
* source RLC

R_R    N00824N005950 1k

L_L     N005950 N006171 10nH

V_V1    N00824 0

+SIN 1 10v 500 hz 0 0 0

C_C     N006171 0 1n
```

**Encadré II.5:** La net liste de SPICE qui représente le circuit de la figure II.3

```

--definition de source de tension
ENTITY source is

END;
ARCHITECTURE behavioral OF voltage is

END;
-- definition de la résistance ENTITY resistor is END resistor
ARCHITECTURE behav of resistor is      ...
END behav;
-- definition de la capacity ENTITY capacitor is      ...
END capacitor;
ARCHITECTURE behav of capacitor is
END behav;
-- definition de l'inductance
ENTITY inductor is
...
END inductor;
ARCHITECTURE behav of inductor is
END behav;
--TEST BENCH
Entity circuit is

End;
Architecture behavioral of circuit is
Terminal n1, n2, n3: ELECTRICAL;

Begin
vsrc: entity voltage    (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);

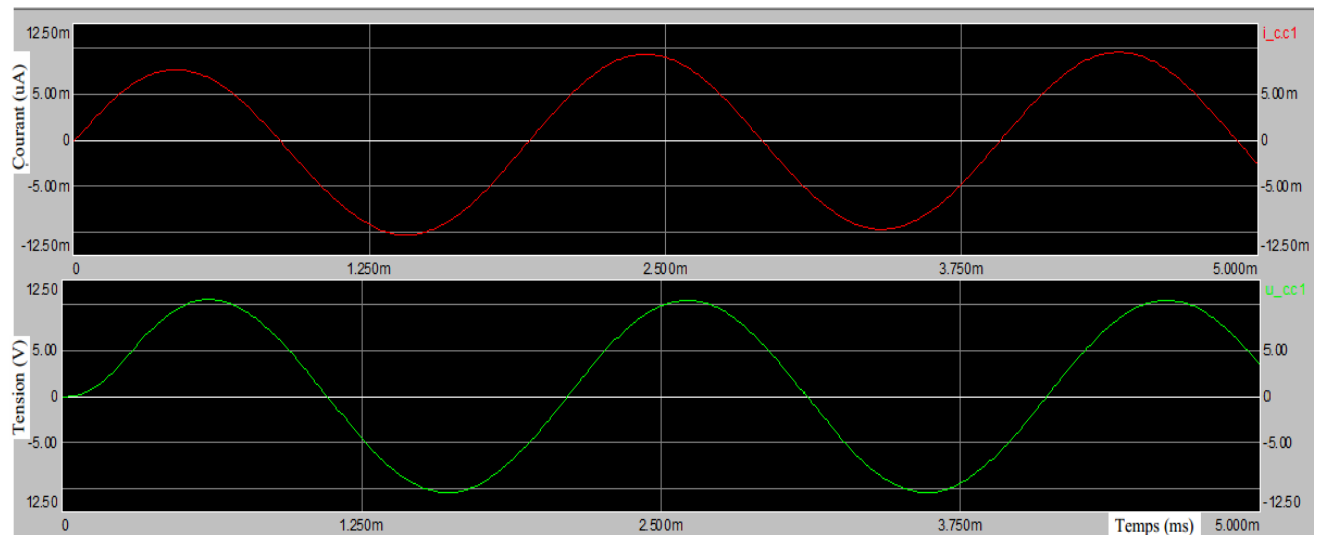
End;

```

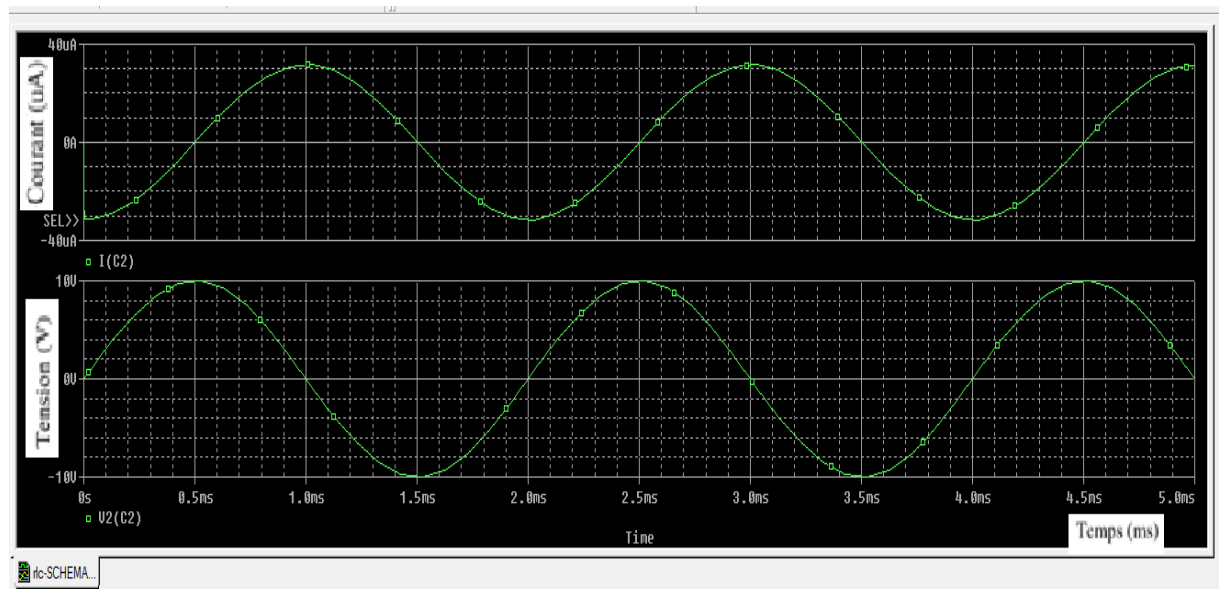
**Encadré II.6: Code VHDL-AMS du le circuit de la figure II.3**

Pour la description en langage VHDL-AMS de ce circuit, nous utilisons les principes de la conception hiérarchique à partir des composants élémentaires résistance, capacité et inductance. Par ailleurs, la définition des paramètres R, C et L comme générique (instruction Genséric), c'est à dire fixés seulement lors de la simulation dans un composant de hiérarchie supérieure, permet d'obtenir une description générique et réutilisable.





**Figure II.4(a):** Résultat de simulation par VHDL-AMS d'un circuit RLC série



**Figure II.4.(b) :** Résultat de simulation par PSPICE d'un circuit RLC série.

A partir de ces figures nous pouvons constater que le VHDL-AMS est capable de nous fournir des résultats comparables à ceux donnés par SPICE.

## II.5 Les différents niveaux d'abstraction :

Grâce à la modélisation VHDL-AMS, il est possible d'utiliser trois types de modélisation VHDL-AMS : modélisation fonctionnelle, comportementale, et physique. Nous appliquons ces types de modélisation pour le circuit redresseur suivant :

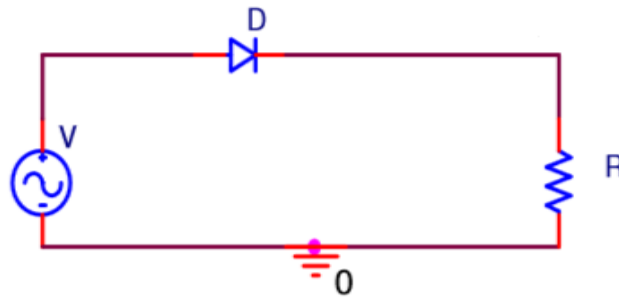


Figure II.5: Circuit de redressement

### II.5.1 Les modèles fonctionnels :

Ceux-ci ne sont pas spécifiques à une technologie particulière. Ils remplissent en quelque sorte une fonction mathématique, valable pour .Ceux-ci ne sont pas spécifiques à une technologie particulière. Ils remplissent en quelque sorte une fonction mathématique, valable pour toute technologie; c'est le schéma bloc de l'automaticien. Ces modèles peuvent être considérés comme des utilitaires, mais nous verrons également qu'ils peuvent servir de base à la spécification fonctionnelle [4]. L'encadré II.7, représente à titre d'exemple, le code VHDL-AMS pour une diode travaillant en redressement mono-alternance (figure II.5)

```

ENTITY Diode IS

END;
ARCHITECTURE functionally OF Diode IS
Constant  iss  : real := 192.1e-12; --parameters de la diode
Constant  rs1  : real := 0.1;
constant  n    : real := 1.0;
constant  vt    : real := 0.0258;
terminal n1,n2: ELECTRICAL;
quantity v_in  across i_out  through n1 TO electrical_ground;
quantity u_D   across i_D    through n1 TO n2;      --diode
quantity u_r1000 across i_r1000 through n2 TO electrical_ground;
BEGIN
v_in==1000.0 * sin (314.0 * now * 10.0); -- The sinusoidal voltage source equation
i_r1000 == u_r1000/0.001;    -- resistor
i_D == iss*(exp ((u_D - rs1*i_D)/(n * vt)) - 1.0); -- diode

END;

```

Encadré II.7: Code VHDL-AMS du modèle fonctionnel de la diode de la figure II.5

### II.5.2 Les modèles comportementaux :

Ces modèles décrivent le comportement du composant par un bloc comportemental. Le rôle du composant dans un circuit donné comme le redressement dans notre exemple. C'est en effet dans un modèle comportemental qui sera décrit des phénomènes de nature diverse mais ayant de fortes interactions:

- Interactions numériques- analogiques dans le domaine électrique.
- Interactions multi-technologiques dont l'étude est basée sur l'analogie avec l'électricité [4].

```

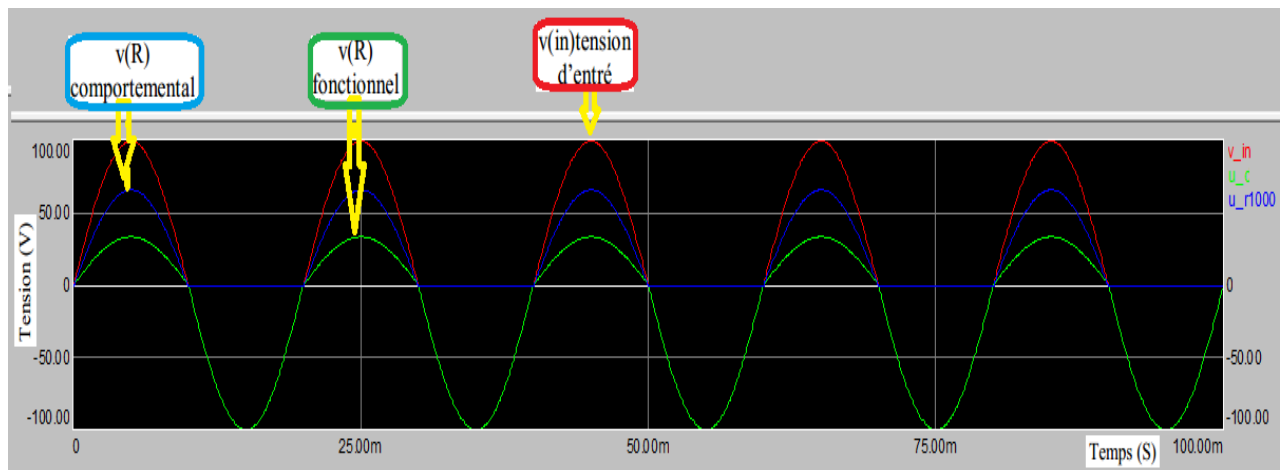
ENTITY Diode IS
END;
ARCHITECTURE behavioral OF Diode IS
Terminal n1, n2: electrical;
Quantity v_in across i_out through n1 TO electrical_ground;
quantity u_D across i_D through n1 TO n2;
Quantity u_r1000 across i_r1000 through n2 TO
electrical_ground;
BEGIN
v_in==1000.0 * sin (314.0 * now * 10.0); -- The sinusoidal voltage
source equation.      i_r1000 == u_r1000/0.001;    -- resistor
If v_in > 0.00 use    -- diode
u_D == 0.0;
else    i_D == 0.0;
End use;
END;

```

**Encadré II.8:** Code VHDL-AMS du modèle comportemental de la diode de la figure II.5

### II.5.3 Les modèles physiques :

Ils se rapprochent des modèles de dispositifs de type SPICE, Ces modèles issus de la description de la physique du système sont analogues aux modèles phénoménologiques. Mais malheureusement ces modèles nécessitent la connaissance de tous les paramètres internes et externes du composant. Ceci reste une limite sérieuse pour ce type de modèles [4].



**Figure II.6:** Simulation comportementale et fonctionnelle de la diode en redressement.

Nous remarquons que l'amplitude du signal redressé dans le cas de la modélisation comportementale est plus supérieure que celui de la modélisation fonctionnelle. Cette différence est due à la chute de tension qui existe aux bornes de la résistance interne de la diode. En effet, dans la modélisation comportementale nous avons considéré la diode comme un interrupteur parfait donc de résistance interne nulle.

## II.6 Conclusion :

Dans ce chapitre nous avons présenté deux types de modélisation : SPICE et VHDL-AMS. Nous nous apercevons que SPICE ne peut pas répondre aux besoins de la multi-technologie et la multi-abstraction d'une manière simple.

Nous portons notre choix pour le reste de notre travail sur le langage VHDL-AMS. Nous avons alors présenté des exemples de simulation de composants discrets sous ce même langage. La différence entre les niveaux d'abstraction (fonctionnelle et comportementale) existant est illustrée sur l'exemple de la diode en redressement.