



Modélisation à haut niveau d'abstraction de l'intégrité du signal dans les bus de communication

Ruomin Wang

► To cite this version:

Ruomin Wang. Modélisation à haut niveau d'abstraction de l'intégrité du signal dans les bus de communication. Mécanique [physics]. Université Pierre et Marie Curie - Paris VI, 2014. Français. ⟨NNT : 2014PA066196⟩. ⟨tel-01081284⟩

HAL Id: tel-01081284

<https://theses.hal.science/tel-01081284v1>

Submitted on 7 Nov 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



HAL Authorization

**THÈSE DE DOCTORAT DE
L'UNIVERSITÉ PIERRE ET MARIE CURIE**

Spécialité
Électronique

École Doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Ruomin WANG

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

**Modélisation à haut niveau d'abstraction de l'intégrité du
signal dans les bus de communication**

soutenance prévue le 15 juillet 2014

devant le jury composé de :

M. Francis CALMON	Rapporteur
M. Luc HÉBRARD	Rapporteur
Mme. Sonia BEN DHIA	Examineur
M. Marc HÉLIER	Examineur
M. Jacques-Olivier KLEIN	Examineur
M. Patrick GARDA	Directeur de thèse
M. Julien DENOULET	Co-encadrant
M. Sylvain FERUGLIO	Co-encadrant
M. Farouk VALLETTE	Invité

Laboratoire d'informatique de Paris 6 - LIP6
4 place Jussieu
75 252 Paris cedex 05

*La science est faite d'erreurs, mais
d'erreurs qu'il est bon de commettre, car
elles mènent peu à peu à la vérité.*
—Jules Verne (1828-1905)

Remerciements

Je souhaite tout d'abord remercier grandement mon directeur de thèse, M. Patrick Garda, pour la confiance qu'il m'a témoignée en acceptant de diriger ma thèse et de m'avoir accueilli au sein de l'équipe SYEL.

Je tiens à remercier M. Julien Denoulet, M. Sylvain Feruglio et M. Farouk Vallette d'avoir co-encadré ce travail de thèse. Ils n'ont eu de cesse de m'encourager et de me soutenir durant ces trois années et demie. Les soutiens qu'ils m'ont apportés dans la conduite des travaux et dans la rédaction de ma thèse ont vraiment été indispensables.

Je remercie vivement M. Francis Calmon et M. Luc Hébrard d'avoir accepté d'être les rapporteurs de ces travaux. Je suis également très honoré que Mme. Sonia Ben Dhia, M. Marc Hélier et M. Jacques-Olivier Klein aient accepté de faire partie de mon jury.

Je souhaite aussi remercier tous les membres passés et présents de l'équipe SYEL du LIP6. Je voudrais remercier plus particulièrement M. Sylvain Viateur (qui simplifie la vie de tout le monde, en plus de préparer des crêpes délicieuses), M. Bertrand Granado (qui m'a beaucoup aidé pour préparer le dossier de soutenance), M. Olivier Romain (qui m'a donné de précieux conseils au début de ma thèse) et M. Andrea Pinna (qui m'a encouragé pendant la rédaction de mon manuscrit). Je souhaite également remercier les personnels administratifs qui ont géré l'équipe SYEL ces dernières années.

Je n'oublie pas les collègues actuels et anciens avec qui j'ai partagé le même bureau : Mehdi, Julien, Brunel, Cédric, Wilfried, les 2 Imen, Alexandre, Christian, Laurent, Mohammed et Syed, avec qui j'ai partagé les bons et mauvais moments de ces années de thèse.

Je remercie également tous les amis avec qui j'ai partagé ma vie en France : Yu, Xue, Xiyang, Qiaoqiao, Yilin, Ye, Qiang, Jizheng et Weidong.

Enfin, je souhaite remercier ma famille en Chine qui m'a soutenu et, plus particulièrement, mes parents. Il m'est impossible de trouver des mots pour dire à quel point je les aime.

Pour conclure, je souhaite bien évidemment remercier mon extraordinaire femme Yuhui (et notre petit amour grandissant en elle) qui m'accompagne depuis 9 ans et sans qui rien n'aurait été possible.

Résumé

Résumé

En raison de l'évolution technologique vers les petites dimensions, de la densité d'intégration croissante et de l'augmentation des vitesses de fonctionnement, l'analyse de l'intégrité du signal est devenue de plus en plus critique dans la conception des systèmes électroniques. Plusieurs méthodes d'analyse ont été proposées et sont utilisées par les concepteurs. Cependant, en considérant l'hétérogénéité croissante des systèmes et la réduction du temps de mise sur le marché des applications, les concepteurs ont besoin de nouvelles méthodes travaillant à haut niveau d'abstraction, afin qu'elles puissent être intégrées facilement à un modèle au niveau système de l'application, et ainsi analyser l'intégrité du signal au plus tôt dans le cycle de conception. Dans cette thèse, nous proposons une méthode de modélisation de l'intégrité du signal basée sur deux types de blocs complémentaires, nommés blocs fonctionnels et blocs non-fonctionnels, décrits à l'aide d'un même langage (C/C++ et les bibliothèques SystemC/SystemC-AMS), et donc aisément simulables dans un unique environnement. Les blocs fonctionnels servent à modéliser les comportements idéaux du système. Les comportements non-idéaux, introduits par les problèmes d'intégrité du signal, sont modélisés dans les blocs non-fonctionnels à l'aide de réseaux de neurones. Pour valider notre méthodologie, deux applications autour des bus I²C et USB 3.0 ont été modélisées à l'aide de notre méthode. Les résultats de simulations démontrent la faisabilité de notre méthodologie. En la comparant à des modèles de référence, notre méthode permet de réduire de façon remarquable le temps de simulation (20% par rapport à un modèle équivalent SystemC/SystemC-AMS à bas niveau et 99% par rapport à un modèle SPICE équivalent). En termes de précision, l'écart moyen entre nos modèles et les modèles de référence est d'environ 3%. Notre méthode offre enfin certaines possibilités de flexibilité et de modularité. Il est en effet possible de paramétrer les modèles (par exemple, pour émuler les modes de configuration d'un émetteur) et de composer un modèle d'une application complexe à l'aide de sous-modèles élémentaires. En perspective, cette méthode originale de modélisation de l'intégrité du signal à haut niveau d'abstraction pourra être intégrée au futur environnement de conception de systèmes cyber-physiques.

Mots-clefs

Intégrité du signal, Système hétérogène, Modélisation à haut niveau, Réseau de neurones, SystemC-AMS, Système cyber-physique.

Abstract

The continuing growth of electronic technology has led to smaller size, higher density of integration and higher frequencies. As a result, signal integrity analysis becomes nowadays a more and more critical challenge in the electronic system design process. To address this issue, designers have proposed and used several approaches. However, in respect of the higher heterogeneity of modern applications, along with an aggressive reduction of time-to-market, a new modeling methodology is required to provide the system's signal integrity performance at a high-level of abstraction. Moreover, it should be easily interoperable with a functional model of this system. The aim of this work is to propose a new modeling methodology for signal integrity analysis that can meet these requirements. Our method is based on the combination of two kinds of blocks, named functional blocks and non-functional blocks. They are built in C/C++ or SystemC/SystemC-AMS, and can therefore be easily simulated in a single environment. The functional block is used to model the ideal behavior of the system. The non-functional block is used to represent the highly nonlinear and non-ideal behaviors, caused by signal integrity issues. In the non-functional block, neural networks are used to model these non-ideal behaviors. To validate our method, we developed two applications based on I²C application and USB 3.0. Our method greatly increases simulation speed (running 20% faster than a low-level SystemC/SystemC-AMS equivalent model, or 99% faster than a SPICE equivalent model). Furthermore, our method achieves very good accuracy (relative absolute error is around 3%). Finally, our method is a flexible approach, since our models can be parameterized, to help designers to configure their systems. It's also modular: a model of a complex application can be built by combining models of each component. In the future, this original method for high-level modeling of signal integrity could be integrated in the forthcoming design flows of cyber-physical systems.

Keywords

Signal integrity, Heterogeneous system, High-level modeling, Neural network, SystemC-AMS, Cyber-physical system.

Table des matières

Introduction	1
1 Contexte et problématique	5
1.1 Introduction	5
1.2 Systèmes hétérogènes	5
1.2.1 Définition	5
1.2.2 Défis de la modélisation d'un système hétérogène	7
1.3 Intégrité du signal	9
1.3.1 Principaux phénomènes détériorant le signal utile	9
1.3.2 Défis pour l'analyse de l'intégrité du signal	12
1.4 Conclusion et problématique	14
2 Etat de l'Art	17
2.1 Introduction	17
2.2 Méthodes de modélisation des systèmes hétérogènes	18
2.2.1 Processus de conception et niveaux d'abstraction	18
2.2.2 Langages de conception	20
2.2.3 Bilan et conclusion	25
2.3 Méthodes de modélisation de l'intégrité du signal	26
2.3.1 Principales techniques de simulation de l'intégrité du signal	26
2.3.2 Outils de modélisation d'intégrité du signal	27
2.3.3 Bilan et conclusion	32
2.4 Conclusion	33
3 Modélisation	35
3.1 Introduction	35
3.2 Présentation des outils	37

3.2.1	SystemC-AMS	37
3.2.2	Réseaux de neurones artificiels	44
3.2.3	Conclusion	52
3.3	Blocs fonctionnels	52
3.4	Blocs non-fonctionnels	53
3.4.1	Introduction	53
3.4.2	Etape 1 - Acquisition des couples d'entrées/sorties-désirées	54
3.4.3	Etape 2 - Apprentissage du réseau de neurones	55
3.4.4	Etape 3 - Construction du modèle SystemC-AMS TDF implémentant le réseau de neurones et les adaptateurs	57
3.4.5	Conclusion	61
3.5	Conclusion	61
4	Simulation et Résultats	63
4.1	Introduction	64
4.2	Première Application : système communiquant via un bus I ² C	64
4.2.1	Protocole I ² C	64
4.2.2	Présentation du système à modéliser et du modèle de référence	67
4.2.3	Développement de notre modèle	70
4.2.4	Résultats de simulation	75
4.2.5	Conclusion	79
4.3	Deuxième Application : système communiquant via un bus USB 3.0	80
4.3.1	Protocole USB 3.0	81
4.3.2	Présentation du système à modéliser et du modèle de référence	84
4.3.3	Développement de notre modèle	88
4.3.4	Résultats de simulation	94
4.3.5	Conclusion	103
4.4	Conclusion	103
5	Conclusion et Perspectives	105
5.1	Conclusion	105
5.2	Perspectives	106
	Publications	109

Table des figures	111
Liste des tableaux	115
Bibliographie	117

Introduction

Les premiers circuits intégrés ont été développés pour des applications militaires et spatiales au début des années 1960. Depuis, ils sont omniprésents dans notre vie quotidienne. L'industrie des circuits intégrés est l'une de celles qui a le plus rapidement évolué dans toute l'histoire des technologies. Tous les paramètres caractérisant l'électronique varient de façon exponentielle : la densité des transistors double chaque 18 à 24 mois ; les dimensions des transistors sont divisées par 2 tous les 3 ans environ ; la fréquence maximale des transactions double tous les 4 ans [BC11]. Ces évolutions remarquables ont permis aux concepteurs de développer des systèmes de plus en plus complexes, possédant des performances et fonctionnalités extraordinaires, pour une multiplicité d'applications (communications, implants médicaux, véhicules, etc.), comme l'illustre la figure 1 où l'on peut observer trois circuits réalisés à des époques différentes.

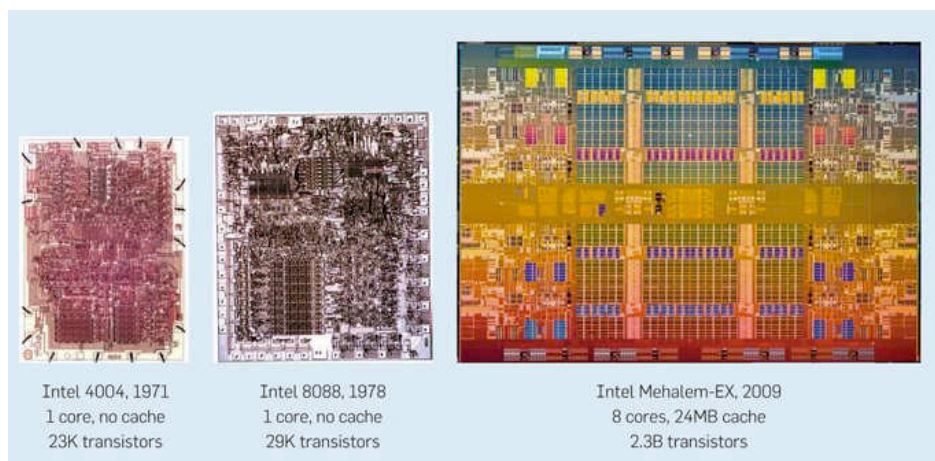


FIGURE 1 – Exemple de trois microprocesseurs d'Intel réalisés à des époques différentes [BC11].

Aujourd'hui, les systèmes électroniques sont connectés à tous les domaines (mécanique, thermique, automatique, etc.), notamment dans les systèmes embarqués ou les systèmes cyber-physiques [BG11]. De tels systèmes sont dits hétérogènes. Cette hétérogénéité conduit nécessairement à développer de nouvelles méthodes de conception qui permettent la co-conception des blocs appartenant à différents domaines (par exemple, les fonctions numériques et analogiques, l'électronique et l'informatique, l'optique et l'électronique d'un capteur d'images).

Les produits d'électronique grand public, comme les téléphones mobiles ou les ordinateurs, sont devenus de plus en plus populaires. En conséquence, les communications et

les échanges de données entre ces produits se sont fortement accrus. Non seulement la fréquence d'utilisation croît, mais également le volume des données à transmettre. Par exemple, les utilisateurs de téléphones mobiles envoient non seulement des textos, mais aussi des photographies et de la vidéo, à l'aide du WiFi ou du réseau 3G/4G. Le transfert et le traitement de ces données volumineuses entre les machines demandent des protocoles de communications filaires ou sans fils à grande vitesse. Dans le tableau 1, nous donnons à titre d'exemple quelques protocoles de communications. Cependant, à grande vitesse, de nombreux phénomènes peuvent dégrader le signal. Aussi, l'analyse de l'intégrité du signal, qui était négligée pour des applications basse fréquence, est devenue indispensable et doit dorénavant être considérée par les concepteurs dès le démarrage d'un projet.

TABLE 1 – Exemples de protocoles de communication à grande vitesse.

Protocole	Vitesse
WiFi (IEEE 802.11ac)	1 Gbit/s
USB 3.0	10 Gbit/s
PCI Express	16 Gbit/s
HDMI 2.0	18 Gbit/s
100 Gigabit Ethernet	100 Gbit/s
InfiniBand 12x	24 Gbit/s à 300 Gbit/s

Parallèlement à ces évolutions technologiques, les impératifs économiques actuels demandent que tout nouveau circuit soit mis sur le marché dans des délais de plus en plus courts. Cependant, les différentes parties d'un système complexe sont généralement modélisées et conçues séparément par différentes équipes de concepteurs, et à l'aide de différentes méthodes. De ce fait, cela conduit souvent à des problèmes de fonctionnement lorsque les différents blocs sont finalement associés. Quant aux problèmes d'intégrité du signal, ils ne sont souvent pris en compte qu'au stade du prototypage matériel. Si un problème est détecté, il faut alors refaire toutes les étapes de modélisation et de prototypage. Cette analyse tardive, d'un point de vue modélisation, fait que le nombre de paramètres à intégrer dans les modèles est généralement important, ce qui rend leur construction complexe et chronophage. De ce fait, le temps de mise à disposition sur le marché (*time-to-market*) et le coût de développement sont accrus.

Il est donc nécessaire de développer pour les systèmes hétérogènes une méthode permettant de modéliser conjointement les performances liées à l'intégrité du signal et les fonctionnalités "idéales" du système. Afin d'être efficace pour des applications complexes, cette méthode doit impérativement s'appuyer sur des modèles à haut niveau d'abstraction, ce qui peut néanmoins potentiellement dégrader la précision des résultats simulés. Dans cette thèse, nous allons adresser ces problématiques en proposant une méthode innovante de modélisation de l'intégrité du signal.

Dans le chapitre un, nous nous attacherons, d'une part, à montrer l'évolution des systèmes électroniques au cours de ces dernières années, en particulier l'évolution vers l'hétérogénéité. D'autre part, nous présenterons les principaux phénomènes générant des problèmes d'intégrité du signal. A la fin de ce chapitre, nous préciserons les difficultés engendrées par ces deux points et exposerons plus en détail la problématique et les challenges

de nos travaux.

Dans le chapitre deux, nous résumerons les langages et les méthodes existant pour la modélisation des systèmes hétérogènes et la prédiction des performances d'intégrité du signal. Nous préciserons également en quoi elles répondent ou non à la problématique soulevée.

Dans le chapitre trois, nous présenterons notre méthodologie originale. Dans un premier temps, nous préciserons les outils nécessaires à la réalisation de notre méthode. Puis nous exposerons la méthode elle-même. Cette méthodologie est basée sur la combinaison de deux modèles complémentaires : un bloc fonctionnel et un bloc non-fonctionnel. Dans le bloc fonctionnel, ce sont les fonctionnalités du système qui sont modélisés. Dans le bloc non-fonctionnel, ce sont les problèmes liés à l'intégrité du signal qui sont pris en compte à haut niveau d'abstraction. Nous présenterons dans ce chapitre la manière de concevoir ces deux blocs.

Dans le chapitre quatre, nous validerons notre méthodologie. Pour ce faire, deux cas d'étude seront présentés : un système communiquant via un bus I²C et un autre via un bus USB 3.0. Nos résultats seront comparés à des modèles de référence de ces mêmes applications.

Enfin, dans le dernier chapitre, nous conclurons nos travaux et donnerons quelques perspectives.

Chapitre 1

Contexte et problématique

Sommaire

1.1	Introduction	5
1.2	Systèmes hétérogènes	5
1.2.1	Définition	5
1.2.2	Défis de la modélisation d'un système hétérogène	7
1.3	Intégrité du signal	9
1.3.1	Principaux phénomènes détériorant le signal utile	9
1.3.2	Défis pour l'analyse de l'intégrité du signal	12
1.4	Conclusion et problématique	14

1.1 Introduction

Dans ce chapitre, nous présentons le contexte dans lequel s'inscrit la problématique de ce travail de thèse.

Nous présentons d'abord les systèmes hétérogènes. Ensuite, nous exposons les principaux problèmes rendant le signal ou l'information non intègre. Enfin, nous concluons en présentant la problématique soulevée dans nos travaux.

1.2 Systèmes hétérogènes

Dans cette section, nous donnons la définition d'un système hétérogène et présentons les défis que de tels systèmes engendrent.

1.2.1 Définition

Un système électronique moderne est aujourd'hui composé de parties matérielles et logicielles. Ces blocs matériels appartiennent à différents domaines de la physique (électrique, mécanique, optique, thermique, etc.). Par extension, on parle dorénavant de système hétérogène, puisqu'on y retrouve des capteurs et/ou des actionneurs, qui sont de différents

domaines, auxquels on associe des traitements (analogiques, numériques ou même logiciels), une fois que l'information à traiter est dans le domaine électrique.

Dans la figure 1.1, un contrôle automatique de la pression des pneus (*Tire Pressure Monitoring System*, TPMS) d'un véhicule est présenté. C'est un exemple typique de système hétérogène. Ce système contient des capteurs de pression et de température. Les capteurs envoient les informations à l'ordinateur de bord du véhicule en combinant transmission par RF et bus filaire de communication. Ensuite, les informations sont analysées, généralement par des processeurs, et un message est finalement transmis à l'utilisateur sur un écran ou tableau de bord en fonction du résultat de cette analyse.

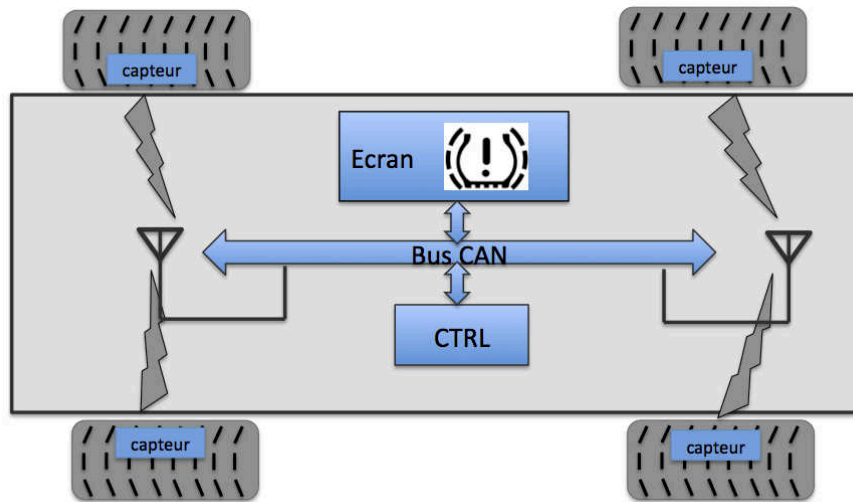


FIGURE 1.1 – Schéma de principe d'un système de contrôle automatique de l'état des pneus d'un véhicule.

Un système hétérogène contient donc de nombreux matériels de différents types, auxquels s'adjoint du logiciel, pour réaliser des fonctionnalités plus ou moins complexes de manière coopérative. Du point de vue de la composition du système, l'hétérogénéité des systèmes peut être présentée sous différentes formes. Selon une vision *top-down*, nous pouvons citer :

- L'hétérogénéité matérielle/logicielle : les tâches peuvent être effectuées par un module matériel dédié ou par le chargement et l'exécution par un processeur d'une application logicielle.
- L'hétérogénéité architecturale : une grande variété d'unités de calcul sont disponibles (DSP, GPU, FPGA, etc.). L'emploi de l'une par rapport à l'autre et leurs associations conduisent à une grande hétérogénéité architecturale. Ainsi, deux exemples d'architecture sont présentés dans la figure 1.2. La figure 1.2.a présente une architecture basée sur un GPU qui interagit avec un processeur. La figure 1.2.b présente une architecture basée sur un FPGA, couplé à un processeur.
- L'hétérogénéité des domaines physiques : les systèmes électroniques modernes font souvent interagir différents phénomènes physiques (électromagnétisme, chimie, biologie, mécanique, optique, etc.). De plus, différents matériaux (silicium, III-V, polymères, etc.) ou techniques d'intégration (empilés, collage, etc.) [OCCH07] sont employés.

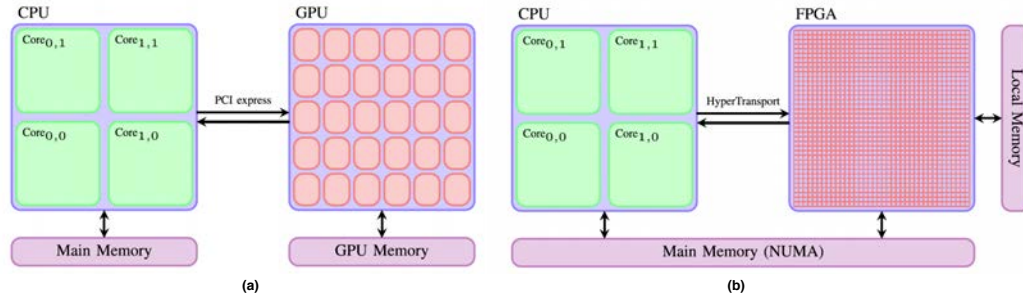


FIGURE 1.2 – Hétérogénéité architecturale : (a) Processeur + GPU et (b) Processeur + FPGA.[BDH⁺10]

- L’hétérogénéité au niveau électrique : cette hétérogénéité se réfère à la coexistence de signaux électroniques différents, suivant la manière dont on les observe et les interprète. Ils peuvent être numériques, analogiques ou mixtes (*Analog and Mixed-Signal*, AMS).

Ces différentes hétérogénéités peuvent coexister au sein d’un même système. Par exemple, dans le TPMS de la figure 1.1, il y a hétérogénéité électrique (AMS/RF), hétérogénéité multi-physique (la pression, le thermique, et l’électrique), et aussi hétérogénéité matérielle/logicielle (via les logiciels embarqués dans l’ordinateur de bord du véhicule). L’association et l’interaction de ces domaines ajoute un niveau de complexité à la maîtrise du système global.

Dans notre thèse, nous nous concentrons sur l’hétérogénéité matérielle/logicielle et sur l’hétérogénéité électrique (AMS/numérique). Un système est présenté dans la figure 1.3 pour illustrer ce cas. Il contient plusieurs sous-blocs de nature différente, comme un microcontrôleur, un processeur, des capteurs, de la mémoire, etc. Ces différents blocs sont connectés ensemble par un canal de communication, (bus I²C, CAN, USB, HDMI, ou autre) auquel chaque bloc accède à l’aide d’un contrôleur d’entrée/sortie. Dans le cadre de cette thèse, bien que les blocs matériels (comme le capteur ou le moteur dans la figure 1.3) puissent se référer à des domaines physiques différents, nous les considérerons comme des composants AMS et non dans leur domaine physique, de plus bas niveau. D’autre part, nous ne discuterons pas des différents types d’unités de calcul. Un microcontrôleur ou un DSP, par exemple, seront traités simplement comme un circuit numérique.

1.2.2 Défis de la modélisation d’un système hétérogène

Les systèmes hétérogènes sont devenus très populaires grâce aux nombreuses fonctions qu’ils permettent de réaliser. Ces fonctions sont toutes gérées par l’électronique, leur modélisation et leur conception sont un challenge en raison de leur complexité.

Dans un flot traditionnel de conception, illustré par la figure 1.4, les spécifications globales sont d’abord définies et un partitionnement en différents domaines (logiciel, matériel numérique, matériel analogique, etc.) est fait. Ensuite, les différentes parties sont modélisées et simulées séparément par les différentes équipes, avec des outils et des méthodes qui leur sont propres. Après avoir vérifié indépendamment chaque partie, on intègre l’ensemble du système et l’on effectue une vérification globale. Si cette vérification (cf. section 2.2.1) échoue, il est nécessaire de revenir en arrière.

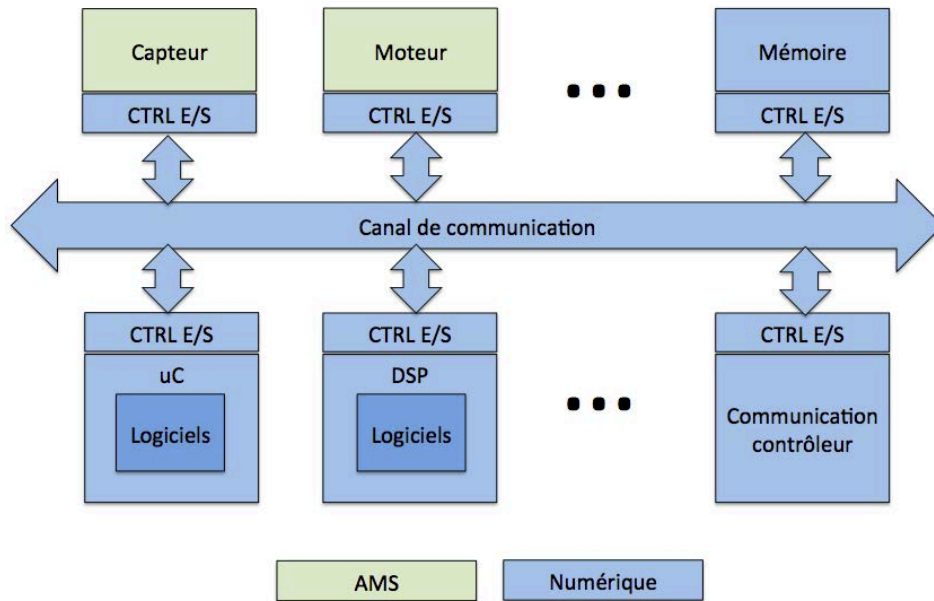


FIGURE 1.3 – Système hétérogène matériel/logiciel.

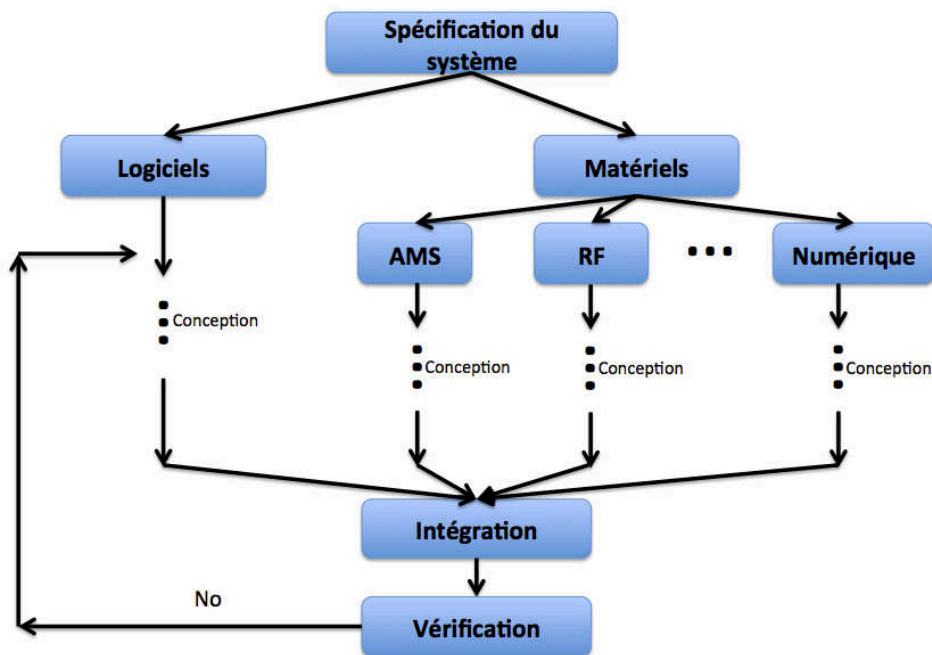


FIGURE 1.4 – Flot traditionnel de conception.

Bien que ce flot ait été adopté et suivi depuis de nombreuses années, il n'est plus adapté à la conception des systèmes actuels qui sont de plus en plus complexes. Ses inconvénients les plus évidents sont :

- Plusieurs équipes travaillent séparément sur différents sous-systèmes. Il y a peu, voir pas, de communication entre les équipes. Les modèles des différentes équipes ne sont donc pas forcément compatibles ou interopérables.

- L'intégration des différents modèles ensemble est rendue difficile pour les raisons suivantes :

 1. Multiplicité des outils : A chaque domaine va correspondre des outils ou langages différents. Par exemple, le langage C/C++ pour le logiciel, VHDL pour un circuit numérique, SPICE pour un circuit analogique.
 2. Multiplicité des niveaux d'abstraction : Les différents domaines sont modélisés à différents niveaux d'abstraction, plus ou moins finement, en fonction du besoin de l'application. Par exemple, les circuits numériques sont souvent modélisés au niveau système, pour une simulation plus rapide ; les circuits analogiques sont souvent modélisés au niveau circuit, pour obtenir une bonne précision.
 3. Multiplicité des modèles de calcul : Différents modèles de calcul sont appliqués pour simuler les différents sous-systèmes (flot de données, événements discrets, temps discret/temps continu, etc.).

Pour aider le concepteur à surmonter ces difficultés, de nouvelles méthodes pour concevoir des systèmes hétérogènes sont indispensables. Ces méthodes doivent permettre, dans l'idéal, de modéliser les différentes parties d'un système hétérogène dans un environnement unifié. De plus, afin de réduire le coût et aussi raccourcir le temps de mise à disposition sur le marché, il serait important que cette méthode soit réalisée à un haut niveau d'abstraction, pour permettre de modéliser les systèmes hétérogènes plus efficacement. Dans notre thèse, nous allons proposer une méthode pour satisfaire à cette demande dans le cadre de l'étude de l'intégrité du signal.

1.3 Intégrité du signal

Dans cette section, nous discutons des problèmes liés à l'intégrité du signal.

1.3.1 Principaux phénomènes détériorant le signal utile

On dit qu'un signal est intègre si les informations qu'il contient ne sont pas dégradées ou transformées lors de son traitement ou sa transmission. L'analyse de l'intégrité du signal est décrite comme la procédure de détection et de correction de ces problèmes d'intégrité afin qu'ils ne perturbent pas les fonctionnalités du système.

Comme illustré dans la figure 1.5.a, dans un système numérique, le signal est transmis d'un composant à un autre sous forme logique ("1" ou "0"). Cependant, ces signaux sont en réalité analogiques. La forme réelle d'un signal numérique est plus proche de ce qui est présenté dans la figure 1.5.b. Le seuil haut (V_{ih}) et le seuil bas (V_{il}) sont définis par les concepteurs ou la technologie employée. Si le signal reçu est au-dessus du seuil haut, il est interprété comme un "1" logique. Si le signal reçu est au-dessous du seuil bas, il est interprété comme un "0". Pour s'assurer que les informations sont transmises correctement, il faut que le signal soit toujours au delà des seuils haut ou bas. La forme du signal n'étant pas parfaite, en raison de dégradations introduites par les interconnexions ou l'environnement extérieur, il est possible d'aboutir à une mauvaise interprétation de sa valeur.

Quand les systèmes électroniques travaillaient encore à quelques dizaines de mégahertz, avec des temps de transition de quelques dizaines de nanosecondes, l'intégrité du

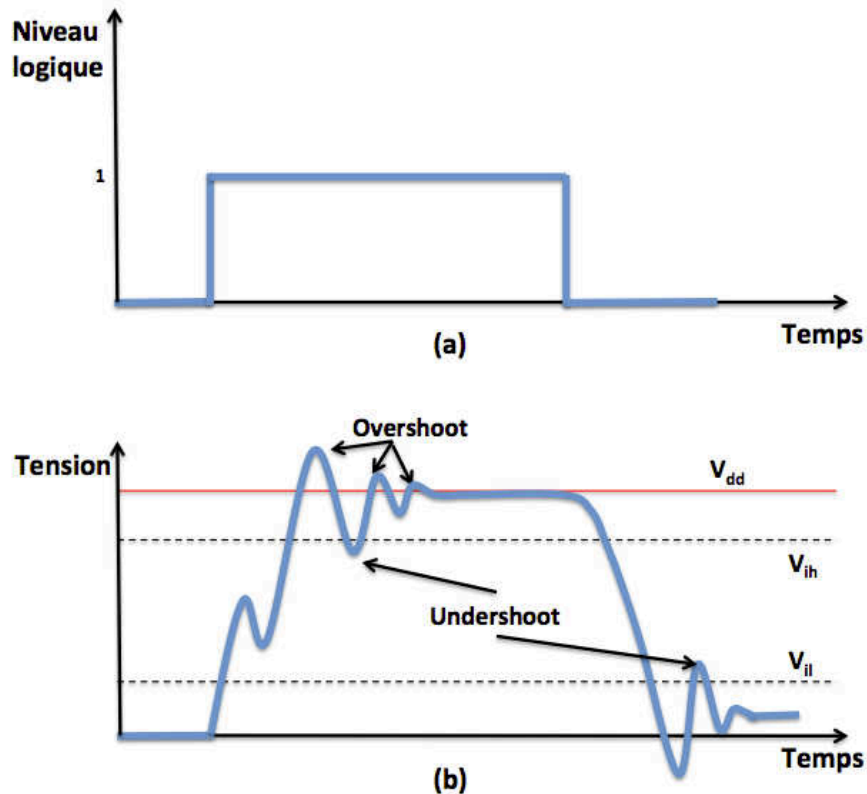


FIGURE 1.5 – (a) Forme idéale d’un signal numérique et (b) Forme réelle de ce même signal.

signal n’était pas un problème majeur dans le processus de conception. Pour cette raison, les concepteurs ne s’en souciaient pas ou peu. Cependant, au fil des évolutions technologiques, les fréquences d’horloges se sont accrues et les temps de transition ont diminué. Les problèmes d’intégrité du signal sont alors devenus de plus en plus importants, pouvant conduire au dysfonctionnement ou même à l’endommagement du système. Par conséquent, dans la réalisation d’un système électronique rapide, la phase d’analyse de l’intégrité du signal est désormais indispensable.

Dans cette phase d’analyse, deux aspects des signaux sont à considérer :

- Le temps : est ce que le signal atteint sa destination au moment adéquat ? Reste-t-il stable suffisamment longtemps ?
- La qualité du signal : est-il visible et interprétable correctement (amplitude du signal) ?

Le premier aspect est d’autant plus critique que les signaux sont de plus en plus rapides. Le problème du temps est souvent causé par la longueur du chemin que le signal doit parcourir entre deux éléments, et également par les caractéristiques de l’interconnexion entre ces deux éléments (ligne de transmission). Par exemple, l’échantillonnage de données est généralement déclenché sur un front montant ou un front descendant d’un signal d’horloge. De plus, pour s’assurer que l’échantillonnage soit correct, on demande que le signal transmis reste stable avant et après l’échantillonnage (*setup time* et *hold time*). Cependant, à cause des problèmes de délai de propagation sur les lignes de transmission, ou encore

des problèmes de bruit (gigue temporelle), le signal n'arrive pas toujours à respecter ces temps, comme l'illustre la figure 1.6, dans le cas d'une violation du *setup time*. Le système peut alors entrer dans des états métastables.

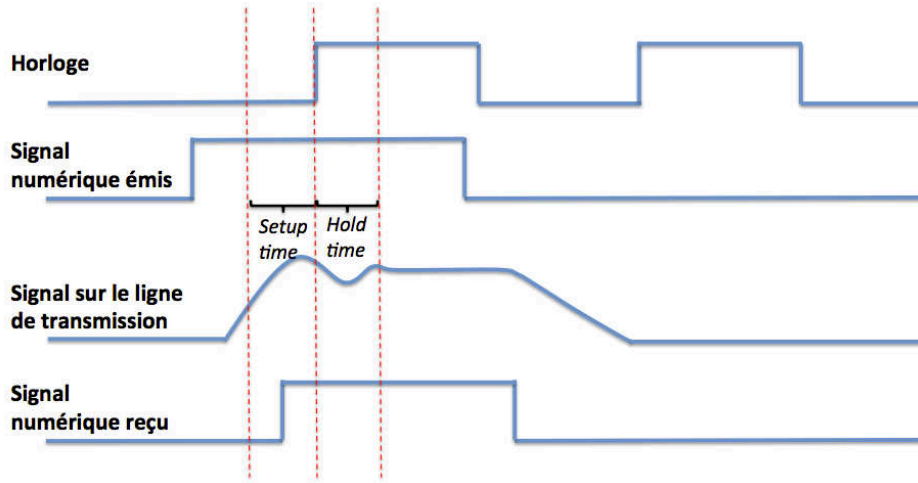


FIGURE 1.6 – Illustration de l'influence du délai de propagation.

Le deuxième aspect concerne la qualité du signal. La distorsion d'un signal peut être causée par différents problèmes, comme la diaphonie (capacitive ou inductive), le franchissement de seuil, le régime transitoire oscillant (*undershoot/overshoot*), comme l'illustre la figure 1.5.b.

Les problèmes liés à l'intégrité du signal sont multiples. On peut les classer en quatre types principaux [Bog03] :

- Mauvaise adaptation d'impédance [Bro03] : d'un émetteur vers un récepteur, le signal traverse plusieurs composants comme les plots d'E/S, la ligne de transmission, etc. Si le signal rencontre une rupture d'impédance le long du chemin de transmission (phénomène de désadaptation d'impédance), une partie de celui-ci est réfléchi vers l'émetteur. La figure 1.7 présente ce phénomène de désadaptation et les équations 1.1 et 1.2 définissent l'amplitude du signal réfléchi :

$$V_{\text{réfléchi}} = \rho \cdot V_{\text{incidente}} \quad (1.1)$$

$$\rho = \frac{Z_1 - Z_0}{Z_1 + Z_0} \quad (1.2)$$

avec ρ le coefficient de réflexion, et Z_0 et Z_1 les impédances des différents composants du système (par exemple, une ligne de transmission et un connecteur) ou des différentes couches d'un PCB. Ces phénomènes de réflexion causent une déformation du signal (*undershoot* et *overshoot*).

- La diaphonie entre deux ou plusieurs éléments [SH00] : les pistes/connexions voisines influent mutuellement les unes sur les autres, en raison des champs électromagnétiques qu'elles rayonnent. Cela est généralement modélisé par des capacités ou des inductances parasites et mutuelles. Par exemple, dans la figure 1.8.a, les deux pistes métalliques adjacentes sont déposées sur un diélectrique et sont liées par une capacité mutuelle C_m (couplage capacitif) et par une inductance mutuelle L_m (couplage

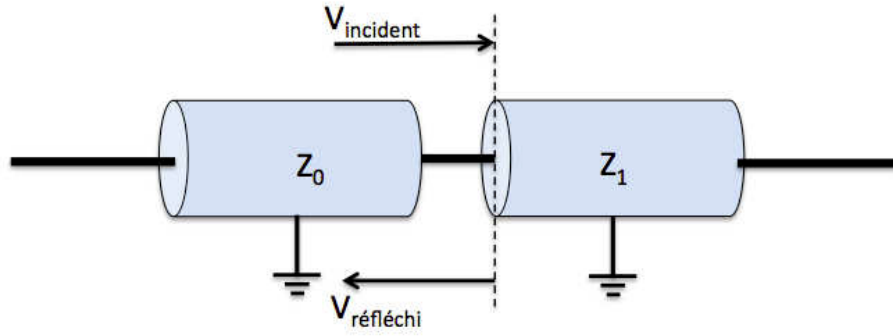


FIGURE 1.7 – Illustration du phénomène de désadaptation d'impédance.

inductif) (cf. figure 1.8.b). Chaque transition de l'agresseur (piste en rouge) perturbe la victime (piste en bleu). Le courant injecté sur la victime est alors défini par l'équation 1.3 :

$$I_c = C_m \cdot \frac{dV}{dt}. \quad (1.3)$$

La tension induite sur la victime est alors définie par l'équation 1.4 :

$$V_i = L_m \cdot \frac{dI}{dt}. \quad (1.4)$$

- Les variations des potentiels de références (phénomènes de rebond des potentiels de masse, *IR drop*) [You01] : Dans un système électronique, l'alimentation et la masse sont normalement communes à tous les composants. Cependant, ces plans d'alimentation et de masse ne peuvent souvent pas être directement connectés aux composants. Ils sont souvent reliés par des interconnexions (au minimum, les fils de bondings). Les courants traversant ces éléments vont donc créer une variation de tension due aux impédances de ces connexions. Par exemple, dans la figure 1.9, les inverseurs A et B ont un plan d'alimentation/masse commun. La sortie de l'inverseur A doit normalement rester à "0". Cependant, quand l'entrée de l'inverseur B bascule de l'état "0" à l'état "1", un courant est généré dans le plan de masse. Alors, un saut de tension survient en sortie de l'inverseur A.
- La compatibilité électromagnétique [Pau06] : tout conducteur traversé par un courant crée un champ magnétique. De même, la présence d'une tension dans un conducteur crée un champ électrique. Ces champs interagissent et peuvent créer des troubles sur les autres champs environnants (par exemple, les composants voisins). Les champs électromagnétiques peuvent provenir de l'intérieur d'un circuit ou de l'extérieur. En effet, les systèmes électriques ne sont pas isolés de leur environnement, et sont souvent influencés par les champs électromagnétiques créés par les autres systèmes à proximité.

1.3.2 Défis pour l'analyse de l'intégrité du signal

Comme nous l'avons présenté ci-dessus, l'analyse de l'intégrité du signal n'est pas une discipline nouvelle. Il s'agit finalement de la prise en compte de phénomènes analogiques, microondes et électromagnétiques appliqués au monde du numérique. Elle est devenue de

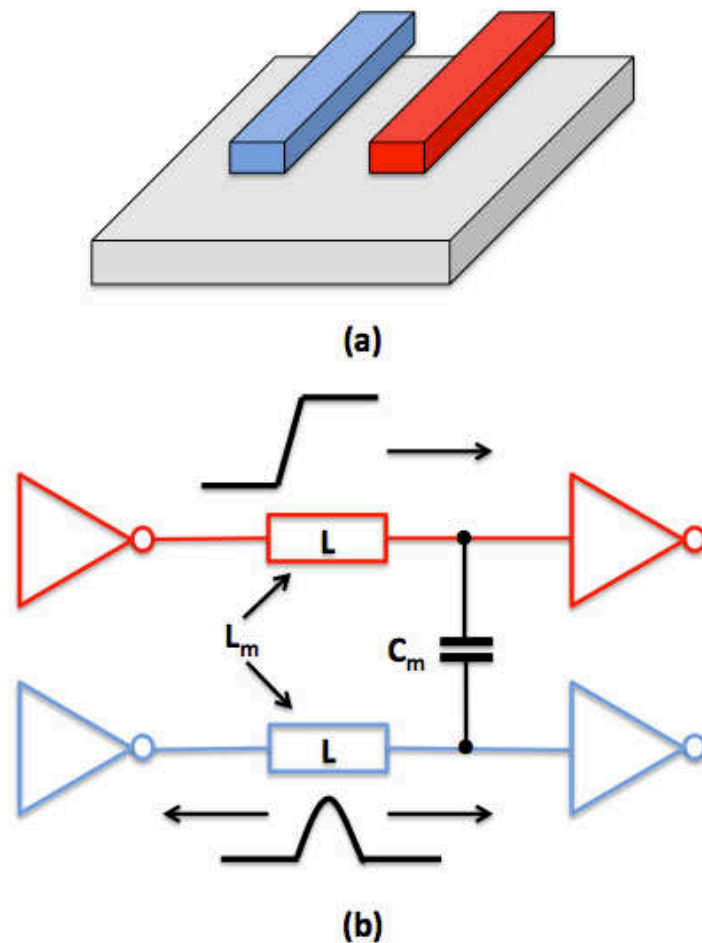


FIGURE 1.8 – Diaphonie entre deux lignes proches.

plus en plus critique dans la conception des circuits modernes à cause de l'évolution des technologies, notamment l'augmentation des fréquences de fonctionnement.

Aujourd'hui, l'analyse de l'intégrité du signal est appliquée à chaque étape du processus de conception. Plusieurs techniques d'analyse de l'intégrité du signal ont été proposées. Dans [Bog03], elle sont classifiées en trois types, selon une approche *bottom-up* :

- Techniques basées sur l'électromagnétisme : l'analyse de l'intégrité du signal s'effectue à l'aide des équations de Maxwell (1D, 2D ou 3D) pour prévoir le champ électromagnétique dans et autour du système et en déduire le comportement réel de celui-ci.
- Approches basées sur les circuits électriques équivalents : ce type de méthode s'appuie sur la résolution d'équations différentielles (basées sur les lois de Kirchhoff) calculées à partir d'un circuit électrique équivalent du système.
- Techniques basées sur le comportement du système : dans ce cas, tout se base sur la réalisation de fonctions de transfert qui représentent les relations entre les entrées et les sorties.

Dans la section 2.3, ces méthodes seront présentées plus en détail. Au fil des ans, de nom-

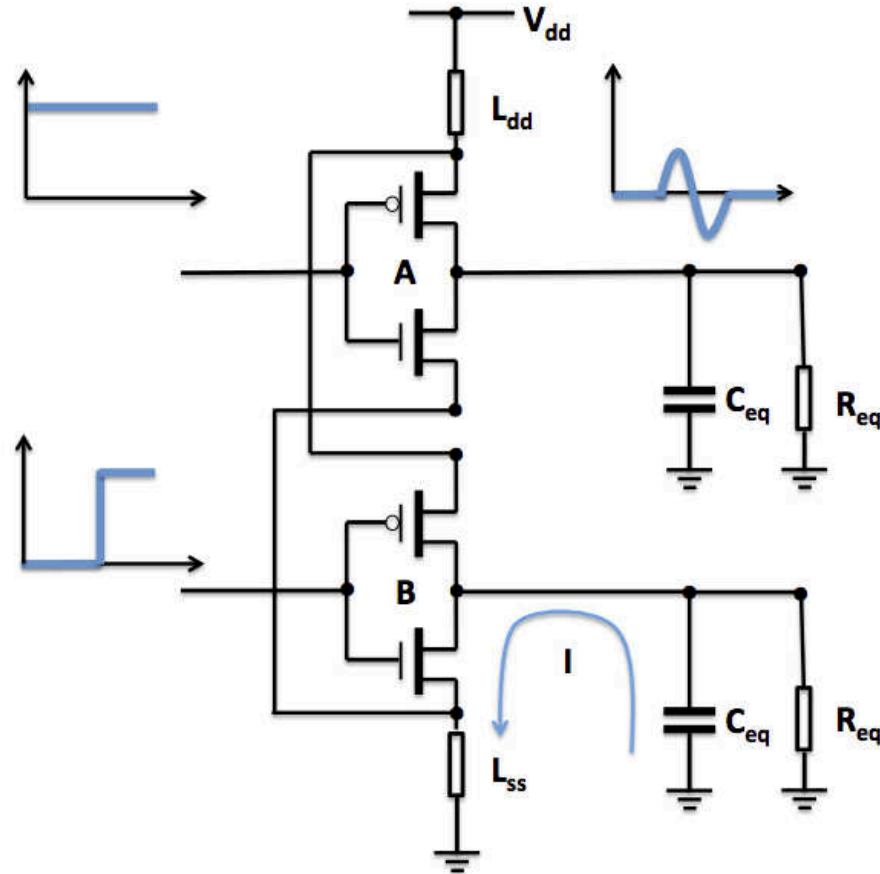


FIGURE 1.9 – Phénomène de rebond du potentiel de masse.

breux outils basés sur ces trois types de solutions ont été proposés et sont utilisés par les industriels et les académiques. Cependant, ils présentent certains inconvénients. Par exemple, la simulation est lourde dans sa mise en œuvre et sa résolution, en particulier quand les systèmes deviennent complexes. La création d'un modèle nécessite de nombreux paramètres. Pour réduire la complexité et raccourcir le temps de réalisation, il est indispensable de disposer de méthodes qui permettent d'analyser l'intégrité du signal dès le début du processus de conception, ce qui implique qu'elles se placent à un haut niveau d'abstraction, puisque les paramètres d'implémentation ne sont pas forcément tous connus à ce stade. D'autre part, cette méthode doit couvrir les différents types de problèmes liés à l'intégrité du signal. Enfin, les modèles conçus à l'aide de cette méthode doivent facilement interagir lors de la simulation avec les modèles fonctionnels idéaux du système.

1.4 Conclusion et problématique

Dans ce chapitre, nous avons commencé par présenter les systèmes hétérogènes au sens large et nous avons raffiné cette définition dans le cadre de notre travail de thèse, à savoir : un système AMS, contenant des parties logicielles et matérielles et échangeant des données via un bus de communication.

En raison de l'évolution des technologies liées à l'électronique, les systèmes sont devenus

de plus en plus hétérogènes. Cependant, cette hétérogénéité conduit aussi à adapter les méthodes de conception (en montant en abstraction et en unifiant les environnements ou les outils de modélisation).

Ensuite, nous avons présenté l'intégrité du signal. Son étude est devenue de plus en plus importante dans le processus de conception des systèmes électroniques. Plusieurs techniques d'analyse de l'intégrité du signal existent. Cependant, elles sont complexes à mettre en œuvre et ne répondent pas à tous les besoins des concepteurs. Il y a donc un intérêt à développer une nouvelle méthode : cette méthode doit se placer à un haut niveau d'abstraction et permettre de modéliser les différents phénomènes dégradant le signal. De plus, une interaction doit être possible entre ces modèles et les modèles fonctionnels du système. En effet, il faut que, naturellement, l'on puisse observer les signaux idéaux, générés par les modèles fonctionnels et y associer une analyse de l'intégrité du signal.

Les travaux présentés dans cette thèse vont donc s'attacher à répondre à ces questions que l'on peut résumer ainsi : comment modéliser, à haut niveau d'abstraction et dans un environnement unifié, les principaux phénomènes d'intégrité du signal, afin qu'ils puissent être associés à une simulation fonctionnelle d'un système hétérogène ?

Chapitre 2

Etat de l'Art

Sommaire

2.1	Introduction	17
2.2	Méthodes de modélisation des systèmes hétérogènes	18
2.2.1	Processus de conception et niveaux d'abstraction	18
2.2.2	Langages de conception	20
2.2.2.1	Langages de description des circuits numériques	20
2.2.2.2	Langages de conception des circuits analogiques	23
2.2.3	Bilan et conclusion	25
2.3	Méthodes de modélisation de l'intégrité du signal	26
2.3.1	Principales techniques de simulation de l'intégrité du signal	26
2.3.2	Outils de modélisation d'intégrité du signal	27
2.3.3	Bilan et conclusion	32
2.4	Conclusion	33

2.1 Introduction

Le chapitre précédent a présenté les évolutions récentes des systèmes électroniques et les problèmes d'intégrité du signal relatifs à ces évolutions. Nous avons indiqué la problématique de nos travaux, à savoir l'apport d'une méthode de modélisation à haut niveau d'abstraction dans un environnement unifié donnant les fonctionnalités d'un système hétérogène et intégrant les principaux phénomènes d'intégrité du signal. Nous sommes intéressés en particulier par les performances de l'intégrité du signal pour des systèmes qui échangent des données sur un bus de communication.

Ce chapitre présente et compare les méthodes existantes et précise en quoi elles répondent ou non à la problématique soulevée. Nous présentons dans un premier temps les principales méthodes actuellement utilisées pour modéliser des systèmes hétérogènes (section 2.2). Puis, dans un deuxième temps (section 2.3), nous décrivons les méthodes usuelles de modélisation de l'intégrité du signal. Nous terminons ce chapitre en reformulant notre problématique, suite aux conclusions apportées par l'état de l'art.

2.2 Méthodes de modélisation des systèmes hétérogènes

2.2.1 Processus de conception et niveaux d'abstraction

Les systèmes hétérogènes sont des systèmes comportant des parties matérielles mixtes (i.e., numériques et analogiques), mais également des parties logicielles intégrées. Les échanges de données se font via des bus de communication. La réalisation de ces systèmes passe par un processus de conception, associé à différents langages de modélisation possibles.

Puisque nos cas d'étude sont des systèmes matériels mixtes (*hardware*, *HW*) et logiciels (*software*, *SW*), nous présentons le processus de conception de ces deux types de systèmes.

Traditionnellement, pour la conception des systèmes *HW/SW*, le système était spécifié conjointement avant d'effectuer un partitionnement de fonctions *HW* et *SW*. Chaque entité était alors développée, simulée et debuggée séparément. On regroupait enfin *HW* et *SW* pour une validation finale [LM93], [BSED07]. Ce processus de conception traditionnel est présenté en figure 2.1.a. Le principal inconvénient de cette approche est que les méthodes de modélisation de ces parties sont totalement différentes. En conséquence, les outils et les modèles utilisés dans les deux branches sont incompatibles. De ce fait, il est difficile voire impossible de les faire interagir au cours du processus de conception. Cela introduit une grande problématique dans l'étape d'intégration *HW/SW*. Ce processus traditionnel demande donc de grandes ressources de développement (outils/hommes/temps) et augmente le temps de mise à disposition sur le marché (*time-to-market*).

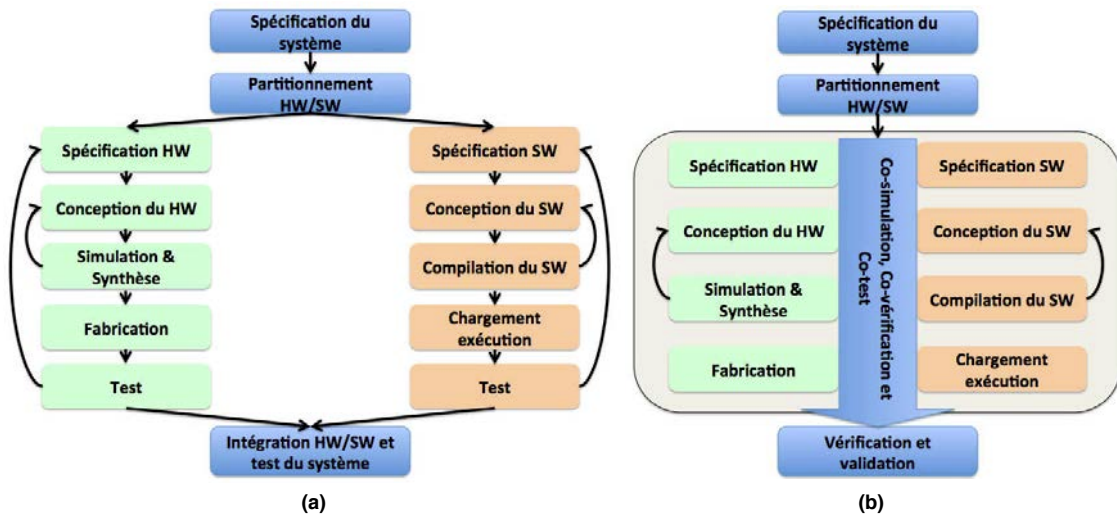


FIGURE 2.1 – (a) Flot de conception *HW/SW* traditionnel. (b) Flot de co-conception *HW/SW*.

Pour résoudre ce problème, les techniques de co-conception (*co-design*) *HW/SW* ont été proposées à partir des années 90. La définition de la co-conception matérielle/logicielle est donnée dans [Sch10] :

"Hardware/Software co-design is the design of cooperating hardware components and software components in a single design effort."

La figure 2.1.b présente un processus standard de co-conception. L'objectif de la co-conception est de concevoir ensemble matériel et logiciel depuis le niveau système [MG97]. Le matériel et le logiciel sont spécifiés, synthétisés, simulés et vérifiés ensemble. Par rapport au processus de conception traditionnel, le processus de co-conception permet de valider et déboguer simultanément le *HW* et le *SW*, tout au long du processus de conception. Il permet alors de réduire le temps de mise à disposition sur le marché. Dans l'article [Tei12], l'auteur rappelle ces évolutions et prévoit le futur de la co-conception *HW/SW*. Il présente les quatre générations de langages de co-conception. La première génération a réussi le bi-partitionnement des fonctions *HW* et *SW*, comme la méthode Vulcan [GDM93] et la méthode Cosyma [EHB93]. La deuxième génération a été proposée pour réaliser la co-simulation, comme le *framework Ptolemy* [EJL⁺03]. La troisième génération s'est concentrée sur le DES (*Design-Space Exploration*) et la co-synthèse pour les systèmes multi-cœurs, comme SystemC [Ope] et SpecC [DGG02]. La co-conception *HW/SW* a ainsi beaucoup progressé. Cependant, les trois générations existantes sont plutôt concentrées sur la co-conception des logiciels et des circuits numériques. Les nouveaux challenges de l'industrie et du milieu académique s'orientent dorénavant vers la co-conception et la co-vérification des composants matériels appartenant à différents domaines comme le numérique, l'analogique, ou la mécanique. C'est l'objectif de la quatrième génération d'outils qui est en cours de développement.

La conception des circuits mixtes (numérique/analogique) connaît des problématiques similaires. Le processus de conception traditionnel se divise en deux parties distinctes : le processus de conception des circuits numériques et le processus de conception des circuits analogiques. Ces deux processus ont des niveaux d'abstraction différents, résumés dans la figure 2.2.

Au cours des dernières décennies, les nouveaux langages ont profondément modifié le domaine de la conception numérique. Ces divers langages (VHDL [vhd], SystemC [Swa01], etc.) permettent de représenter les circuits numériques à différents niveaux d'abstraction, comme nous le verrons dans la section 2.2.2.1.

Pour la conception analogique, la situation est très différente. Le processus de conception analogique ne présente habituellement que trois niveaux d'abstraction [GR00] : le niveau système, le niveau fonctionnel, et le niveau circuit/transistor. Toutefois, la conception analogique au niveau système ou au niveau fonctionnel est aujourd'hui encore un goulet d'étranglement. Depuis la première norme de SPICE [NP73] sortie en 1973, cet outil ainsi que ses dérivés (NGSPICE [NV13], HSPICE [hsp08], Eldo [eld05], etc.) sont les outils les plus populaires et les plus utilisés par les concepteurs analogiques [VGKM06]. Mais pour concevoir des circuits plus grands et plus complexes, ces outils deviennent insuffisants et il est nécessaire de travailler avec des outils de plus haut niveau. Ainsi, si on veut modéliser et simuler conjointement du matériel (numérique et analogique) et du logiciel, un langage de description analogique au niveau système devient indispensable. Pour répondre à cette demande, le *framework Ptolemy* et les extensions AMS (*Analog Mixed Signal*) des langages pour circuits numériques (SystemC-A [AJK05], SystemC-AMS [sys10], etc.) sont proposés, comme nous le verrons dans la section 2.2.2.2.

Les langages de description analogique au niveau système rendent possible la co-conception des circuits analogiques et numériques. Cette conception conjointe permet d'accélérer le processus de conception, tout comme la co-conception *HW/SW* [Tei12, HW01]. Pour réaliser cette co-conception, nous avons besoin de langages de description au niveau système. Le langage C et les *C-based languages* peuvent être utilisés pour réaliser

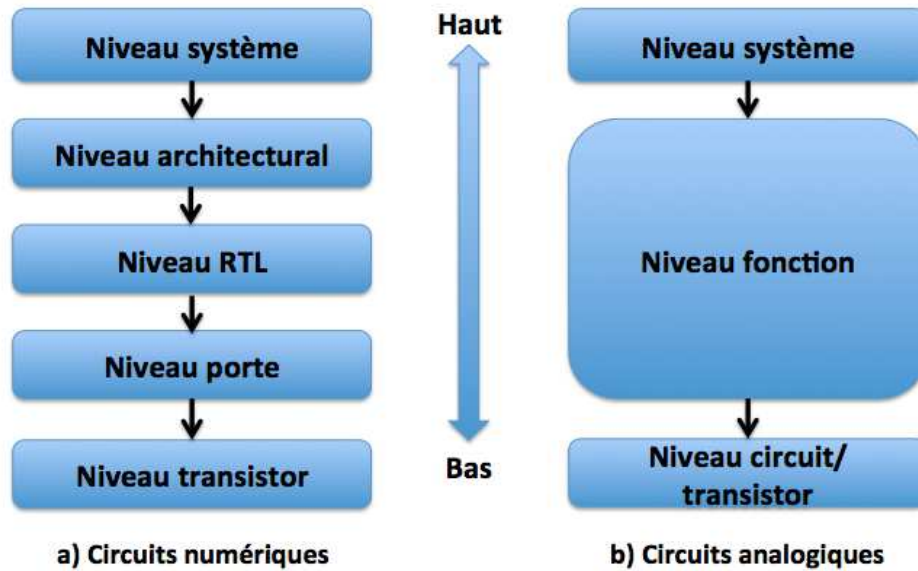


FIGURE 2.2 – Niveaux d'abstraction pour la modélisation de circuits numériques et analogiques

ce processus de co-conception [WH08]. Pour la partie *HW*, les *C-based languages* et les extensions AMS de ces langages permettent de concevoir les circuits numériques et les circuits analogiques au niveau système. Parallèlement, les langages de type C primitif ou C++ permettent naturellement de décrire l'application logicielle et s'intègrent facilement aux *C-based languages* utilisés pour la conception *HW*.

Dans la section suivante, nous allons présenter ces principaux langages de conception matérielle.

2.2.2 Langages de conception

Pour réaliser un processus de co-conception, différents langages de conception sont disponibles. Certains sont spécifiques au *HW*, d'autres au *SW*. Ici, nous nous concentrons sur les langages de conception matérielle car les parties logicielles sont facilement intégrables au processus de co-conception, contrairement au *HW*. Nous réalisons d'abord un état de l'art spécifiquement pour les circuits numériques, puis pour les circuits analogiques.

2.2.2.1 Langages de description des circuits numériques

Pour le matériel numérique, VHDL (*VHSIC Hardware Description Language*) [vhd] et Verilog [ver] sont les langages de description de matériel (*Hardware description language*, HDL) les plus populaires. Le langage VHDL est maintenant le HDL majoritairement utilisé en Europe, alors que Verilog est souvent préféré dans les autres pays.

VHDL a été normalisé en 1987 par l'IEEE, sous la norme IEEE 1076-87. La dernière version IEEE 1076-2008 a été publiée en 2009. Le langage VHDL permet de définir un système par une structure hiérarchique de composants, par son comportement, ou encore par ses caractéristiques temporelles. Ce langage permet d'aller du niveau système jusqu'à

un niveau proche du matériel. Il est néanmoins plus particulièrement adapté au niveau RTL (*Register Transfer Level*) et au niveau porte logique. Il peut l'être aussi au niveau système et au niveau circuit avec quelques limitations. De plus, il peut aussi bien servir à simuler un système qu'à le synthétiser.

Verilog est un autre HDL qui a été développé par la société *Gateway Design Automation* en 1984. A l'origine, il s'agissait d'un langage propriétaire mais il est devenu un standard public en 1995 (IEEE 1364-1995). La dernière version IEEE 1364-2005 a été publiée en 2006. Comparé au langage VHDL, le langage Verilog possède une syntaxe fort différente mais des fonctionnalités équivalentes, même si les deux langages ne couvrent pas exactement les mêmes niveaux d'abstraction (voir figure 2.3) [Smi96].

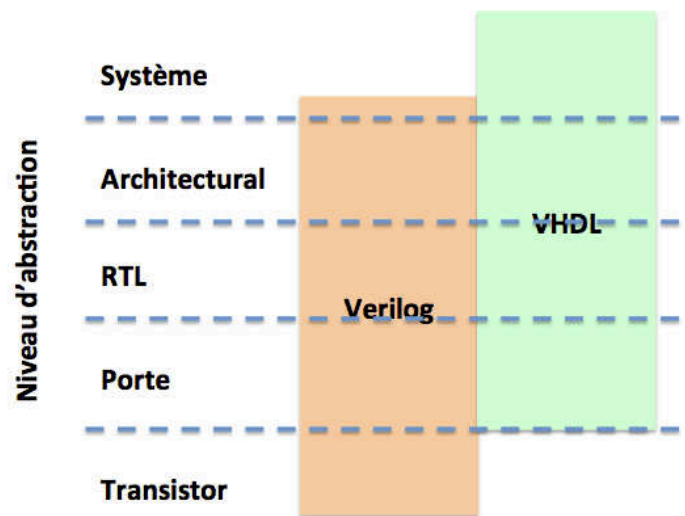


FIGURE 2.3 – Niveaux d'abstraction couverts par les langages VHDL et Verilog.

Une autre approche pour les concepteurs est d'utiliser des langages basés sur C ou C++ pour décrire le matériel au niveau système. Les *C-based languages* ajoutent à la couche C ou C++ des bibliothèques de classes qui permettent de décrire des blocs matériels. Ce type de langage peut donc être compilé par n'importe quel environnement C ou C++ standard. Trois principaux *C-based languages* sont communément employés : Impulse C, SpecC et SystemC.

Impulse C [Imp] est développé par *Impulse Accelerated Technologies* de Kirkland Washington. C'est un langage combiné à une bibliothèque de fonctions compatibles avec le langage C et qui émule la concurrence propre aux systèmes matériels. Il est notamment utilisé pour programmer des applications visant des cibles de type FPGA. Une chaîne d'outils logiciels basés sur Impulse C a été développée par [MRWSS07] pour faciliter l'utilisation de FPGA dans les applications de contrôle. Dans [KS09], une méthodologie de co-conception *HW/SW* sur FPGA s'appuyant sur Impulse C a été présentée.

SpecC [DGG02], cette extension du langage ANSI C, est un langage de description de système qui permet de décrire des applications embarquées au niveau système. Il a été proposé par l'Université Irvine de Californie, en coopération avec plusieurs entreprises comme TOSHIBA et HITACHI. L'organisme STOC (*SpecC Technology Open Consortium*)

se consacre à la promotion du langage SpecC, qui est considéré comme un standard pour décrire des circuits intégrés au Japon depuis 2000. STOC a dévoilé la version 2.0 de SpecC en juin 2002.

SystemC [Ope] a été bâti comme une bibliothèque spécialisée de classes C++. Cette bibliothèque comprend aussi, en standard, un moteur de simulation à événements discrets. SystemC est un ensemble de classes C++ permettant de modéliser non seulement des systèmes matériels (numériques), mais aussi des systèmes mixtes matériels/logiciels. Il est le fruit de contributions de plusieurs sociétés réunies initialement sous le nom d'OSCI (*Open SystemC Initiative*). C'est maintenant *Accellera Systems Initiative* qui est en charge de diffuser, promouvoir et rédiger les spécifications de SystemC. SystemC a été normalisé par l'IEEE en 2005 puis en 2011 (IEEE Std. 1666). Il est, comme VHDL et Verilog, souvent présenté comme un langage de description de matériel car, il peut être utilisé de la spécification jusqu'à la description RTL.

Pour monter en abstraction, SystemC a proposé le standard TLM (*Transaction Level Modeling*). Dans TLM, les communications sont totalement abstraites. La figure 2.4 résume l'architecture de la bibliothèque. Les trois blocs encadrés en pointillés constituent les noyaux de SystemC et les deux blocs au dessus sont les bibliothèques optionnelles. Certains des éléments présentés dans la figure servent à représenter du matériel (modules, ports, signaux, fifo), d'autres à représenter des mécanismes logiciels (sémaphores, mutex), et certains sont suffisamment abstraits pour pouvoir représenter les deux (canaux, événements, interfaces, etc.). Se basant sur SystemC, plusieurs plates-formes de prototypage virtuel ont été proposées, comme SoCLib [soc], OVP [Bai08], CoMET.

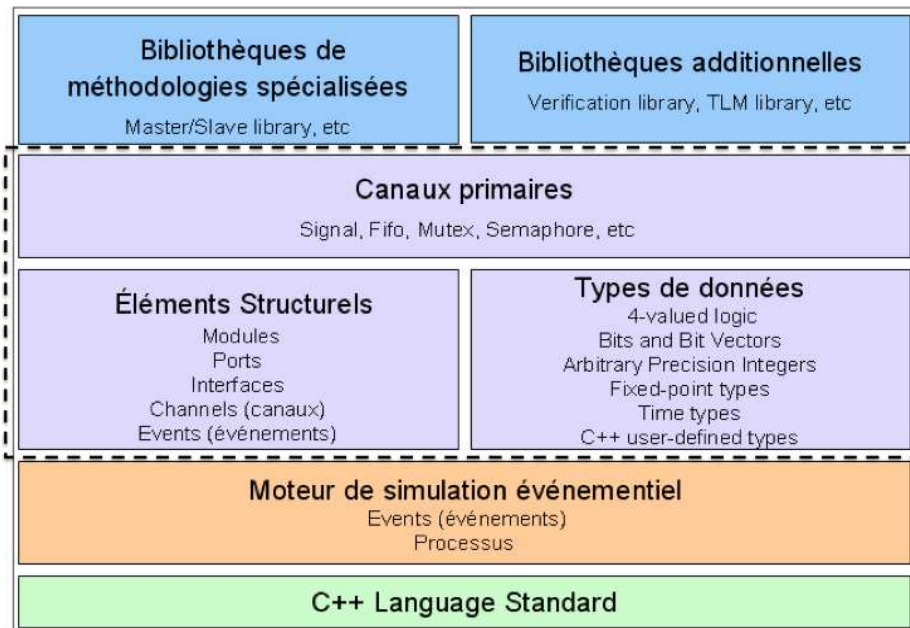


FIGURE 2.4 – Organisation de SystemC.

En parallèle à SystemC, **SystemVerilog** [Acc] s'est imposé comme un langage de description et de vérification des circuits numériques basé sur Verilog. Il est normalisé par l'IEEE Std. 1800. Grâce à la modélisation transactionnelle, un modèle du système peut être obtenu rapidement. Ce modèle transactionnel est disponible bien avant la description matérielle du même système au niveau RTL. Le modèle transactionnel autorise des simu-

lations rapides et permet de construire le logiciel embarqué. Alors, ce modèle permet de s'assurer que le logiciel est bien adapté à l'architecture matérielle et qu'il fonctionne correctement. Quand la plate-forme matérielle de prototypage est prête, le modèle transactionnel devient le modèle de référence pour vérifier l'architecture matérielle. Cette vérification nécessite des vecteurs et un banc de test pour exécuter le modèle RTL. SystemVerilog est particulièrement utilisé pour cette tâche, car il est capable de générer des vecteurs de tests aléatoires et de vérifier automatiquement l'architecture matérielle. Contrairement à SystemC, SystemVerilog est un langage plus fermé. Il est difficile d'intégrer des nouveaux modèles de calcul (*model of computation*, MoC) dans le langage.

Tous les langages cités précédemment sont développés avec l'objectif de décrire à différents niveaux d'abstraction des systèmes numériques. En revanche, ils ne peuvent pas modéliser les parties analogiques.

2.2.2.2 Langages de conception des circuits analogiques

Les simulateurs analogiques mettent en œuvre des systèmes d'équations algébriques et différentielles, linéaires ou non, pour décrire les circuits. Le plus célèbre des simulateurs électriques est **SPICE** [NP73] de l'Université de Berkeley. Dans SPICE, les systèmes sont représentés par des circuits équivalents constitués de composants élémentaires comme des résistances, des condensateurs, des inductances, des sources de courant/tension, etc. SPICE calcule alors les tensions et les courants à partir des équations différentielles associées aux lois de Kirchhoff. Depuis lors, SPICE (et les simulateurs dérivés) est l'outil le plus populaire pour modéliser les circuits analogiques. Aujourd'hui encore, la précision de ce type de simulateur est toujours appréciée et les simulations obtenues sont à la fois une référence et un passage obligé avant toute fabrication de circuit intégré. Malgré tout, l'outil possède quelques limitations : en effet, le modèle des composants à simuler contient des informations et des caractéristiques spécifiques que la plupart des vendeurs ne veulent pas donner aux clients par peur de piratage (*reverse engineering*). Parfois, les vendeurs donnent juste des fichiers cryptés, mais l'utilisation de ces fichiers est limitée car il est impossible de les lire ou de les modifier directement. Par ailleurs, décrire entièrement un circuit sous la forme d'une liste de transistors et de composants passifs est fastidieux, et le temps de simulation d'un circuit complexe peut être très long. Enfin, un handicap insurmontable est que SPICE ne permet pas de modéliser les parties logicielles d'un système hétérogène. Les concepteurs ont donc besoin de langages travaillant à un plus haut niveau d'abstraction.

Les langages de description des systèmes analogiques à haut niveau sont souvent des extensions des langages présentés dans la section précédente. Par exemple, **VHDL-AMS** [CB99] est un dérivé de VHDL. Il permet la description et la simulation de systèmes mixtes. Ce langage supporte la modélisation à différents niveaux d'abstraction dans le domaine électronique ou non-électronique (hydraulique, thermique, etc.). Les circuits analogiques à modéliser sont descriptibles par des systèmes d'équations différentielles et algébriques (*Differential Algebraic Equation*, DAE). VHDL-AMS propose un noyau de simulation analogique pour résoudre ces équations. Les deux noyaux de simulation, analogique et numérique (VHDL), sont indépendants. De même, **Verilog-AMS** est une extension de Verilog [Ope98]. Bien qu'ils permettent de décrire des systèmes matériels mixtes à différents niveaux, ces langages n'ont que des capacités très limitées à coupler cette description matérielle à l'exécution d'un code logiciel embarqué sur un ou plusieurs modèles de processeurs. La solution souvent utilisée est d'associer les modèles de composants analogiques

à des modèles développés dans d'autres langages supportant la simulation de logiciel embarqué. Des co-simulations peuvent alors être effectuées, mais les performances de ces dispositifs (ex., temps de simulation) sont malheureusement limitées [AJK06], [RCG04] à cause des performances du simulateur analogique. De plus, ils n'offrent pas de possibilités consistantes et cohérentes de raffinement du design.

Simulink [Mat] et **Ptolemy II** [EJL⁺03] sont deux outils qui permettent de modéliser et simuler des systèmes mixtes au niveau système. Simulink offre une alternative pour la modélisation fonctionnelle et comportementale, mais il manque de lien direct avec le flot de conception du matériel et il n'offre aucun moyen de simuler le logiciel. Ptolemy II considère un système complexe comme une composition hiérarchique de structures. Chacune des parties élémentaires est modélisée à l'aide du MoC le plus adapté. Ces modèles de calcul sont également appelés modèles exécutables, car ils peuvent être utilisés pour la simulation ou la synthèse. Toutefois, même si Ptolemy II permet de donner le comportement en temps continu d'un système à un haut niveau d'abstraction, il ne permet pas de raffiner la description à un niveau plus bas (par exemple, à un niveau architectural).

Dans le même temps, le développement de langages AMS (*Analog Mixed Signal*) au niveau système devient un autre point important pour répondre aux besoins croissants de conception de systèmes hétérogènes. Compte tenu du succès remporté par SystemC, notamment dans l'industrie, dès le début des années 2000, l'évolution logique est de développer des extensions analogiques à cette bibliothèque. Plusieurs approches existent comme SystemC-A [AJK05], SystemC-WMS [OBC05] et SystemC-AMS [sys10].

Dans [AJK05], SystemC est étendu à **SystemC-A**. Il prend alors en charge les variables analogiques de différents types et les composants analogiques. Tous les composants analogiques d'un circuit/système contribuent à la mise en place d'une matrice de systèmes pour l'analyse nodale modifiée. L'interconnexion des modèles analogiques et numériques est réalisée par des modèles d'interface spécifiques. Par exemple, le convertisseur numérique-analogique est réalisé en utilisant la méthode d'Euler implicite. Le convertisseur analogique-numérique est réalisé en détectant les franchissements de seuils. Dans [AJKW06], SystemC-A est utilisé avec succès pour modéliser un système d'isolation des vibrations de sièges automobiles. Toutefois, il faut modifier le noyau de SystemC pour l'adapter à ces composants analogiques. De plus, la manière de définir les contributions de la matrice du système est très proche du niveau du circuit. Ce n'est donc pas une méthode très appropriée pour les systèmes complexes.

SystemC-WMS est proposé dans [OBC05]. Les différents modules analogiques de SystemC-WMS communiquent entre eux par échange d'énergie au moyen d'ondes par des interfaces nommés canaux d'onde (*wave channel*). Les interfaces des canaux d'onde sont des interfaces analogiques qui peuvent soutenir différents domaines de la physique. Cette méthode peut être employée aussi pour des signaux non-électriques. Par exemple, dans l'article [GCO⁺07], SystemC-WMS est utilisé pour la modélisation physique des effets d'une guitare électrique. Toutefois, la vitesse de simulation est lente. Le temps de simulation dans [OBC05] est cinq fois plus long que le temps de simulation de MATLAB. La raison de cette lenteur est que le simulateur analogique n'autorise pas le simulateur numérique à travailler indépendamment.

SystemC-AMS est un projet mené depuis 2002. L'article [VGE03] publié en 2003 est la première spécification de cette extension. Initialement, SystemC-AMS ne comportait que deux MoCs [VGE05] : le modèle de calcul de type flot de données synchrone (*Synchronous Data Flow*, SDF) qui autorise une construction algorithmique des modèles analogiques,

et le modèle de calcul de type réseau électrique et conservatif. Le modèle de calcul de type flot de signal, non-conservatif, a été ajouté plus tard pour faciliter la description des systèmes asservis contenant des boucles de rétroaction. La normalisation de SystemC-AMS est une mission de l'OSCI menée depuis 2006. La version 1.0 du standard a été publiée par l'OSCI en 2010 [osc10]. Elle propose trois différents MoCs : le TDF (*Timed Data Flow*), le LSF (*Linear Signal Flow*), l'ELN (*Electrical Linear Network*). Ces trois MoCs ne sont pas directement intégrés à SystemC, mais interfacés à l'aide d'une couche intermédiaire (voir figure 2.5). Cette couche permet de synchroniser le domaine analogique de SystemC-AMS avec le domaine numérique de SystemC. SystemC-AMS a déjà été appliqué pour modéliser différents systèmes hétérogènes, par exemple, un réseau de capteurs sans fils dans [VPB⁺08], la communication en champs proche dans [LZL⁺13] ou un capteur CMOS dans [CSS11]. Une évolution introduite dans [BEGTM11] propose d'implémenter dans le moteur TDF un pas de temps ajustable dynamiquement afin d'optimiser les performances de la simulation.

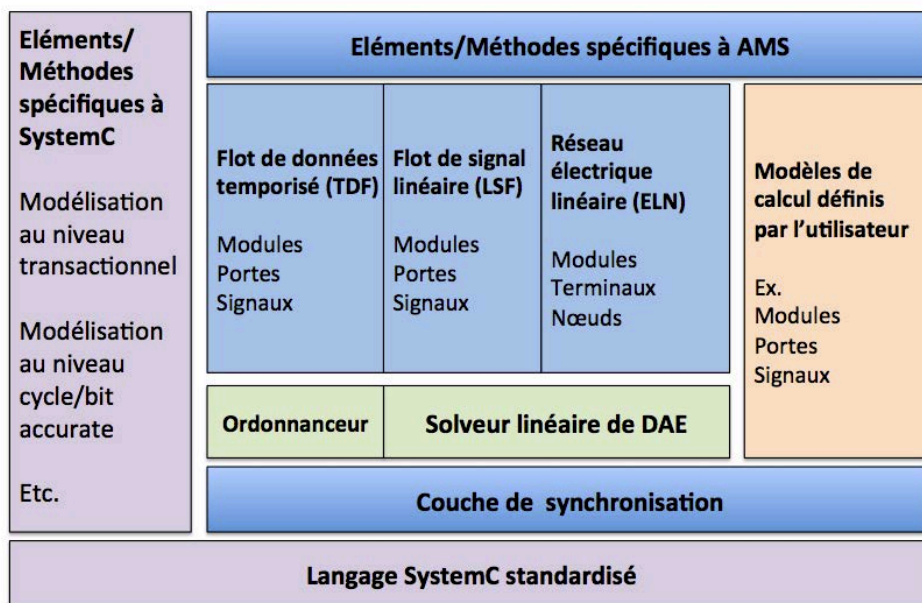


FIGURE 2.5 – Architecture de SystemC-AMS 1.0.

2.2.3 Bilan et conclusion

Cette section a présenté les méthodes pour modéliser les systèmes hétérogènes. Nous distinguons deux processus de conception : un processus de conception pour les systèmes *HW/SW* et un processus pour les systèmes numériques/analogiques. Pour réduire le temps de mise à disposition sur le marché, les processus de co-conception ont été proposés dans les deux cas. Nous avons listé les principaux langages et outils de modélisation des circuits numériques et analogiques. La figure 2.6 présente un comparatif de ces différents langages et indique où ils se situent en termes de niveaux d'abstraction. Parmi ces langages, SystemC-AMS permet de modéliser les systèmes numériques/analogiques au niveau système et il offre la possibilité de modéliser la partie *SW* en s'interfacent facilement avec C++. Il peut donc potentiellement répondre à une partie de notre problématique. Cependant, il manque encore une méthode qui permette de modéliser et simuler les phénomènes

liés à l'intégrité du signal. Ce dernier point sera traité plus en détails dans la prochaine section.

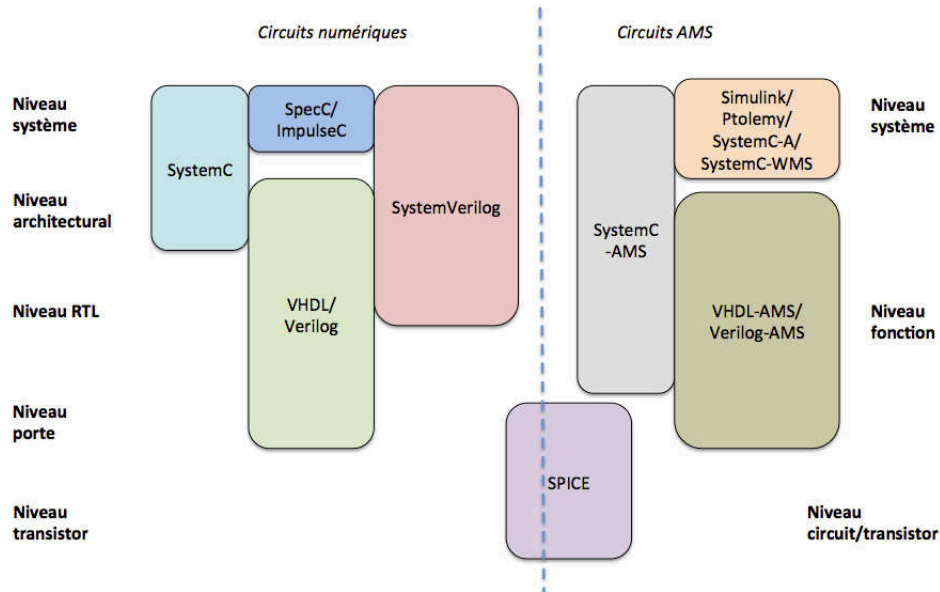


FIGURE 2.6 – Comparatif des langages présentés.

2.3 Méthodes de modélisation de l'intégrité du signal

Après avoir vu les langages de modélisation des systèmes hétérogènes, nous allons, maintenant présenter les méthodes de modélisation de l'intégrité du signal et voir comment elles se positionnent par rapport à notre problématique.

L'intégrité du signal n'est pas un phénomène nouveau. Cependant, tant que les fréquences d'horloges étaient inférieures à quelques dizaines de mégahertz, les concepteurs n'avaient pas besoin de s'inquiéter des effets d'intégrité du signal, et les interconnexions étaient considérées comme idéales. Par contre, dans les systèmes actuels, qui travaillent parfois à des fréquences de quelques gigahertz, les phénomènes de l'intégrité du signal introduisent de nombreux dysfonctionnements. Aussi, de nos jours, la modélisation et la simulation doivent prévoir les effets parasites avec une bonne précision [Bog03]. Un certain nombre de méthodes et d'outils aux niveaux d'abstraction différents ont donc été proposés pour modéliser et simuler l'intégrité du signal. Dans cette section, nous présentons ces méthodes.

2.3.1 Principales techniques de simulation de l'intégrité du signal

Dans le chapitre 1, nous avons montré que la plupart des problèmes d'intégrité du signal sont des problèmes d'électromagnétisme (EM). Il semble donc naturel d'analyser l'intégrité du signal à partir de modèles électromagnétiques. Les simulateurs EM résolvent les équations de Maxwell et donnent les champs électriques et magnétiques à différents endroits du système dans le domaine temporel et/ou fréquentiel. Un phénomène de perturbation (par exemple, un signal électrique) va engendrer une onde électromagnétique, et les

équations de Maxwell vont alors être utilisées pour prédire comment cette onde interagit avec les conducteurs et les diélectriques. Lorsque les champs électriques et magnétiques sont simulés, les interconnexions et les composants passifs doivent être représentés par leurs conducteurs et diélectriques associés avec leurs géométries et les propriétés des matériaux. En effet, les géométries et les propriétés des matériaux définissent les conditions aux limites dans lesquelles ces équations sont résolues. Un certain nombre de simulateurs EM ont été développés et utilisés pour différents systèmes dans [J.Z94], [MD09] et [CJ10]. Les équations de Maxwell prennent en compte tous les phénomènes physiques et les simulateurs EM ont généralement une bonne précision. Cependant, ils nécessitent des algorithmes complexes (2D, 3D) et les temps de simulation sont très longs. Il n'est donc pas raisonnable d'appliquer ce type de méthode sur des systèmes conséquents, tel qu'un MPSoC.

Pour modéliser l'intégrité du signal, il est également possible d'utiliser des simulateurs de circuits électriques de type SPICE. Comparé à un simulateur EM, la précision du simulateur électronique est moins bonne. Cependant, certains problèmes d'intégrité du signal peuvent tout de même être appréhendés correctement par ce type de modèle. Evidemment, il y a des limites à ce qui peut être simulé par ce type de simulateur. Par exemple, on ne peut pas simuler les interférences électromagnétiques.

Il est enfin possible de se baser sur des simulateurs comportementaux. Ils utilisent des tables et des fonctions de transfert spécialisées pour simuler l'évolution des tensions et des courants à partir des relations entre les signaux d'entrées et les signaux de sorties. Alors, les vitesses de simulation sont généralement plus rapides que celles des simulateurs électriques. VHDL-AMS et Verilog-AMS sont des exemples de simulateurs comportementaux.

Le tableau 2.1 compare qualitativement les temps de simulation et la précision des trois types de simulateur. Le simulateur EM a la meilleure performance en précision, mais la pire performance en temps de simulation. Les performances du simulateur comportemental sont inverses. Dans la section suivante, seul le simulateur comportemental et le simulateur de circuit électrique sont présentés plus en détail.

TABLE 2.1 – Techniques de simulation de l'intégrité du signal

	<i>Temps de simulation</i>	<i>Précision</i>
Simulateur EM	+	+++
Simulateur de circuit électrique	++	++
Simulateur comportemental	+++	+

2.3.2 Outils de modélisation d'intégrité du signal

Dans cette section, nous présentons plusieurs approches de modélisation de l'intégrité du signal à l'aide d'un simulateur comportemental ou un simulateur de circuit électrique. La modélisation se réfère ici à la création d'un composant ou d'une fonction qui est utilisée par les simulateurs pour calculer les tensions et les courants aux entrées/sorties d'un circuit et aux nœuds internes de celui-ci. Cette section présente des exemples de ces deux types

de simulateurs en détaillant leur principe de fonctionnement.

D'abord, nous commençons par deux approches classiques :

- **Circuit équivalent** : les simulateurs électriques se basent souvent sur des modèles de circuit équivalent. Par exemple, une ligne de transmission peut être représentée par une combinaison d'éléments R , L et C [You01]. Pour une piste métallique simple, la représentation la plus élémentaire est celle de la figure 2.7.a. Pour des résultats de simulations plus précis, on peut utiliser un modèle en Π (figure 2.7.b) ou en T (figure 2.7.c). Pour représenter certains couplages entre deux lignes de transmission (*crosstalk*), on ajoute des capacités entre les modèles des deux lignes (figure 2.8). On peut trouver dans la littérature différentes approches [Wad91] pour le calcul des valeurs de R , L et C . Ce type de modèle est accepté par les simulateurs de type SPICE (Ngspice, HSPICE, etc).

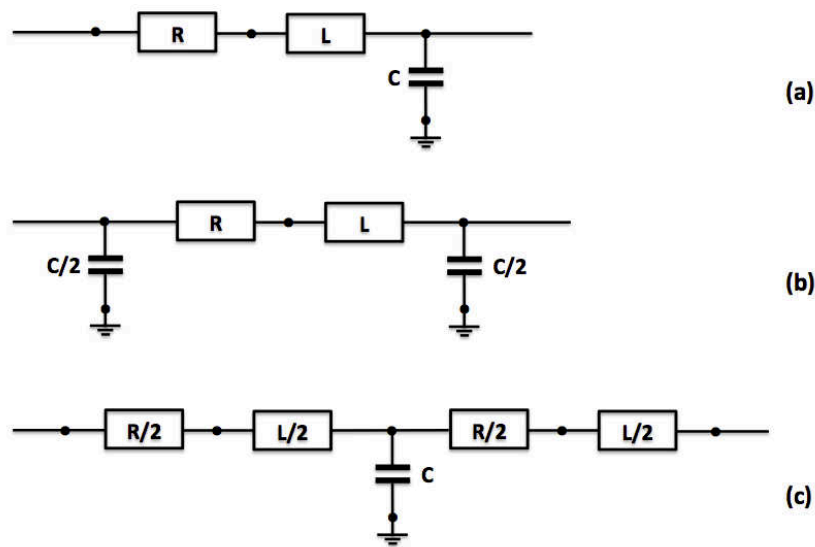


FIGURE 2.7 – Différents types de modèles de ligne de transmission.

- **Paramètres S** : le modèle en paramètres S d'un composant est un type de modèle comportemental. Il donne l'allure des entrées/sorties du composant sous forme matricielle. Bien que le modèle de paramètres S soit applicable pour toutes les fréquences, il est utilisé régulièrement dans le domaine des hyperfréquences. Cependant, il n'est pas adapté à la modélisation de phénomènes non-linéaires.

Certaines approches ont été standardisées. Nous présentons le modèle **IBIS** [ibi13] qui permet de décrire les comportements analogiques des entrées/sorties d'un circuit intégré. Ce standard a émergé au début des années 90. Un modèle IBIS définit la relation entre les entrées/sorties en fonction de leurs courbes caractéristiques statiques (I - V) et dynamiques (V - t). Le standard propose également un modèle de circuit équivalent, qui représente les comportements des courbes, mais sans révéler la structure réelle de l'entrée/sortie (figure 2.9). Dans cette figure, il y a des diodes de protection de l'alimentation et de la masse, la capacité de sortie du composant (C_{comp}) et les composants parasites (L_{pkg} , R_{pkg} , C_{pkg}) du boîtier (*package*). Les composants *Pull-up* et *Pull-down* servent à la modélisation des transitions. Un fichier IBIS doit contenir les informations sur chacun des éléments de ce circuit équivalent. Trois configurations (typique, minimum et maximum) sont données pour

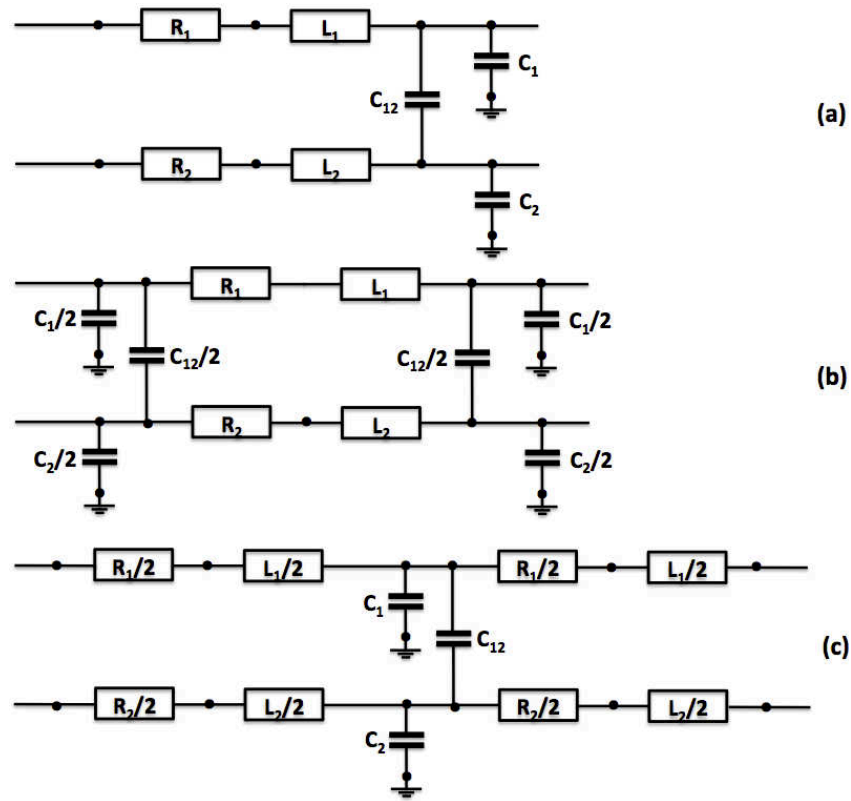


FIGURE 2.8 – Différents types de modèles de deux lignes de transmission couplées.

les valeurs de R , L , C et les tableaux I-V entre $-V_{dd}$ et $2 \times V_{dd}$. L'avantage principal d'IBIS pour un fabricant de circuit intégré est de pouvoir fournir un modèle des entrées/sorties de ses produits sans donner d'informations confidentielles sur les architectures internes. Cependant, la flexibilité d'IBIS est limitée. Un fichier IBIS ne peut être associé qu'à un seul et unique circuit.

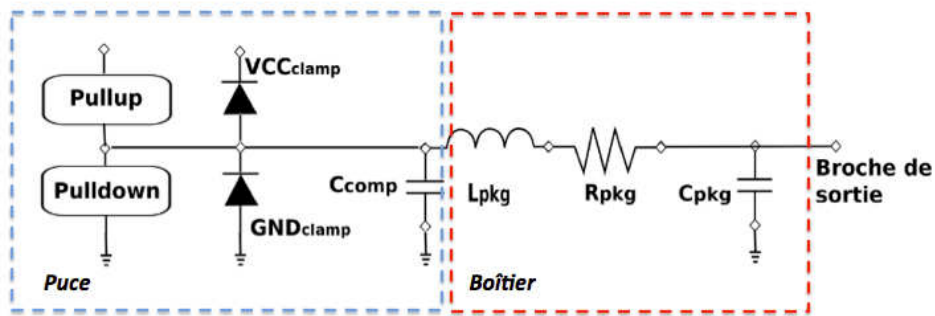


FIGURE 2.9 – Structure d'un modèle IBIS d'une porte de sortie.

Pour améliorer le modèle IBIS, la société JETIA (*Japan Electronics and Information Technology Industries Association*) a proposé le modèle **IMIC** (*Standard for I/O Interface Model for Integrated Circuit*) [imi01] en 2001. Ce modèle est décrit au format SPICE. Par rapport au modèle IBIS, le modèle IMIC permet de caractériser les entrées/sorties d'un circuit intégré. Il permet en outre d'ajouter des informations sur l'inductance mutuelle

et la capacité mutuelle présentes entre les broches du boîtier. La figure 2.10 présente un exemple de modèle IMIC. Le principal avantage du modèle IMIC est la prise en compte de l'influence mutuelle des entrées/sorties et des rails d'alimentation.

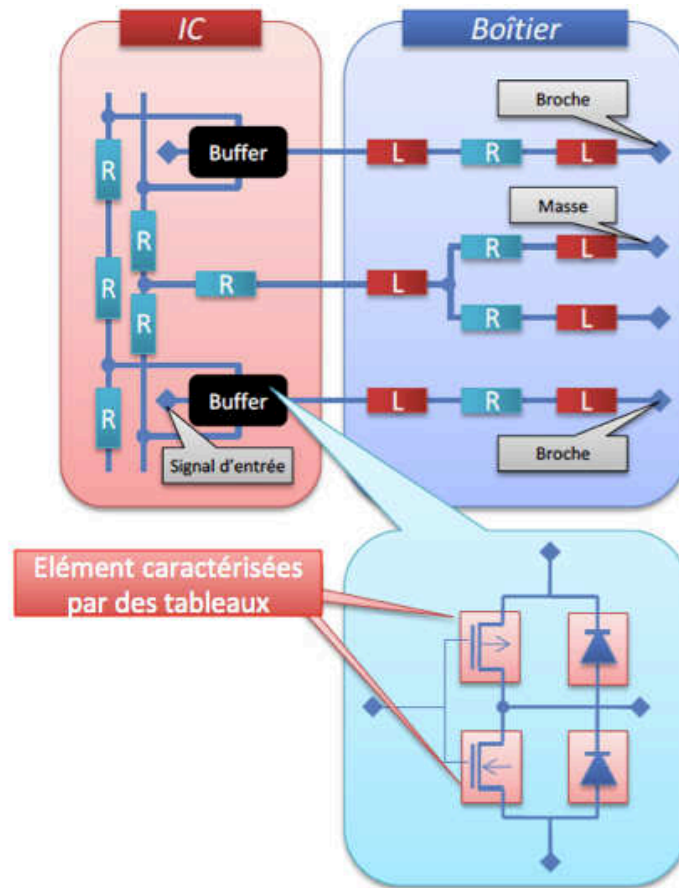


FIGURE 2.10 – Exemple de modèle IMIC.

Le modèle **ICEM** (*Integrated Circuit Electromagnetic Model*) [UTE] est une représentation simplifiée du circuit intégré qui permet d'étudier le champ électromagnétique et de prédire des perturbations sur les rails d'alimentations. La partie fonctionnelle d'un circuit intégré peut être décrite en un ou plusieurs blocs (bloc analogique, bloc numérique, bloc E/S). Le modèle ICEM prend en compte également toutes les connexions internes et externes. Il représente le circuit intégré sous la forme d'un bloc PDN (*Passive Distribution Network*) et d'un bloc IA (*Internal Activity*). La figure 2.11 montre un exemple de modèle ICEM. Le sous-modèle PDN donne un circuit équivalent de la structure électrique des rails d'alimentation pour les différents blocs du circuit et des entrées/sorties. Le sous-modèle IA décrit l'évolution du courant consommé par l'activité interne d'un ou de plusieurs blocs du circuit. Ce sous-modèle peut être représenté par une source de courant. Le bloc interne modélise le couplage parasite à travers le substrat entre les différents signaux de masse. Les perturbations sur les entrées/sorties sont principalement dues au courant consommé dans l'impédance des rails d'alimentation (*IR drop*) et qui vient s'ajouter au courant consommé par le bloc d'entrées/sorties. Pour modéliser ce bloc d'entrées/sorties, le modèle ICEM permet d'utiliser d'autres types de modèles comme le modèle IBIS. Les valeurs des éléments du sous-bloc PDN et la valeur du courant du sous-bloc IA sont déterminées de façon

expérimentale, comme dans [LRP04] et [LDDS⁺08].

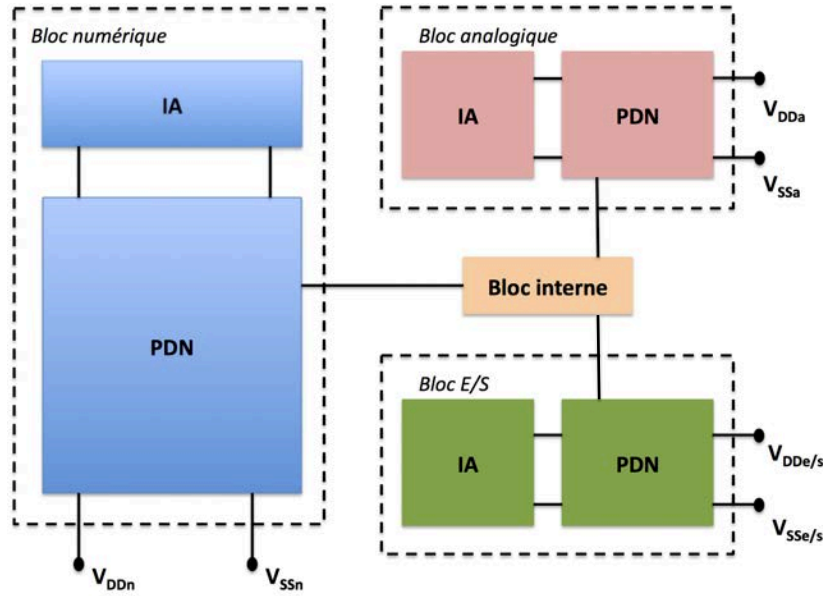


FIGURE 2.11 – Exemple de modèle ICEM.

En plus des méthodes présentées dans les paragraphes précédents, il existe également des approches innovantes. Par exemple, on trouve des méthodes basées sur **les fonctions de transfert spécialisées**. Ce type de méthode calcule généralement les fonctions de transfert par des méthodes mathématiques. Par exemple, dans les articles [ILIF11, SSCM08, CGTMS05, GTHR⁺04, GT03, GTSC03], les auteurs proposent des méthodes pour extraire les fonctions de transfert à partir de mesures. Différents outils mathématiques sont appliqués selon les auteurs (*Local linear state-space* dans [SSCM08], *Hilbert transform* dans [GTHR⁺04], *Time-domain vector fitting* dans [GT03], FDTD dans [GTSC03]). Ces mêmes méthodes sont aussi utilisées pour analyser l'intégrité du signal et la compatibilité électromagnétique dans des composants plus ou moins élémentaires comme les buffers d'entrée/sortie, les lignes de transmission et les dispositifs logiques. Cependant, elles demandent aux concepteurs de manipuler des équations complexes, et la procédure de calcul est donc lourde et coûteuse en temps.

Le **réseau de neurones artificiels** est un autre outil mathématique qui est utilisé pour modéliser l'intégrité du signal. C'est un modèle de calcul inspiré du fonctionnement des neurones biologiques [Hay99]. On apprend dans un premier temps au réseau à résoudre un problème en le paramétrant d'une certaine façon, puis il est utilisé dans le cadre de simulations. Le réseau de neurones a deux principaux avantages : premièrement la possibilité de modéliser les relations non-linéaires et deuxièmement la flexibilité. Il permet ainsi de réaliser des tâches d'analyse, de classification ou de généralisation. Il a déjà été utilisé dans différents domaines [HDB96]. Dans [ZG00], les auteurs montrent le potentiel des réseaux de neurones pour modéliser des circuits électroniques. Dans les articles [VNZ97, WZ98, FYWZ00, CDZ06, CEZ10], les réseaux de neurones sont utilisés pour modéliser différents composants comme un MOSFET, une ligne de transmission, des buffers d'entrées/sorties, etc. Cependant, le choix d'une architecture appropriée de réseau n'est pas trivial. Selon le type de composants ou de systèmes à modéliser, des architectures différentes doivent être choisies par les concepteurs, souvent selon des principes empiriques.

2.3.3 Bilan et conclusion

Cet état de l'art a présenté les trois principaux types de modèles qui permettent de modéliser l'intégrité du signal. Parmi les trois types de modèles, le modèle EM présente la meilleure précision, au détriment d'un temps de simulation important. Pour notre objectif de thèse, un système hétérogène, il est donc trop lourd et coûteux en temps.

Nous avons présenté quelques exemples de modèles circuits et de modèles comportementaux qui sont résumés dans le tableau 2.2. Comparé au modèle EM, le modèle de circuit équivalent est plus rapide. En même temps, les résultats de ce type de modèle présentent aussi une bonne précision. Cependant, pour des circuits et systèmes complexes, là encore, les temps de simulation peuvent être longs. En effet, les simulateurs électriques calculent des tensions et des courants respectant la loi de Kirchhoff. Ils doivent ainsi résoudre des équations différentielles coûteuses en temps de simulation. Par ailleurs, pour déterminer le schéma équivalent le plus approprié, il est nécessaire de connaître de nombreux paramètres. En outre, la flexibilité de ce type de modèle est vraiment limitée et il est difficile, voire impossible d'appliquer un modèle existant à un autre système sans un certain nombre de modifications. Ainsi après mise à jour d'un circuit ou système électronique, il faut reconstruire le modèle.

TABLE 2.2 – Outils pour l'intégrité du signal

Outil	Type
Circuit équivalent	Modèle de circuit
IMIC	
ICEM	
Paramètres S	Modèle comportemental
IBIS	
Fonctions de transfert spécialisées	
Réseau de neurones artificiels	

En comparaison de ces deux précédents modèles, le modèle comportemental présente un certain nombre d'avantages :

- il ne contient pas d'information détaillée sur la technologie de conception et les procédés de fabrication. En fait, le modèle comportemental peut être obtenu à partir des mesures ou des simulations des circuits. Par conséquent, la propriété intellectuelle du fournisseur est protégée.
- son temps de simulation est moindre que celui des autres modèles.

Afin de modéliser l'intégrité du signal à haut niveau d'abstraction, le modèle comportemental est donc le plus adapté à notre objectif. Dans les modèles comportementaux présentés précédemment, le modèle des réseaux de neurones est le modèle qui semble le plus intéressant à ce jour, grâce notamment au fait qu'il permette de modéliser des comportements non-linéaires. Il est aussi très flexible.

2.4 Conclusion

L'état de l'art a présenté en deux parties la modélisation d'un système hétérogène, puis la modélisation de l'intégrité du signal.

Dans la première partie, le processus de conception d'un système hétérogène a été présenté sous la forme d'un processus de conception *HW/SW* et d'un processus de conception numérique/analogique. Ensuite, les principaux langages de conception ont été exposés dans le but de définir quel type de modèle et de langage est le plus adapté à la modélisation à haut niveau d'abstraction de l'intégrité du signal dans les systèmes hétérogènes.

Les méthodes de modélisation de l'intégrité du signal ont été présentées dans la deuxième partie de cet état de l'art. Trois types de modèles d'intégrité du signal ont été exposés dans cette partie : le modèle EM, le modèle circuit et le modèle comportemental.

Parmi les langages de conception présentés, SystemC-AMS avec son environnement unifié semble être le plus approprié à notre problématique car il peut modéliser conjointement au niveau système les parties logicielles et matérielles d'une application. En outre, les trois types de modèles de l'intégrité du signal pourraient être intégrés à un modèle décrit en SystemC-AMS car il supporte les différents niveaux d'abstraction. Cependant, le modèle comportemental de l'intégrité du signal est le plus intéressant par rapport à notre problématique, notamment grâce à la réduction du temps de réalisation du modèle et de la vitesse de simulation. Parmi les modèles comportementaux présentés, le réseau de neurones est le type de modélisation le plus intéressant pour nous. Grâce à la phase d'apprentissage, il nécessite moins de connaissance à priori des éléments à modéliser. Il est le modèle le plus flexible. De plus, il permet de modéliser des fonctions non-linéaires, chose indispensable pour l'ensemble des phénomènes dynamiques liés à l'intégrité du signal.

Dans le chapitre suivant, nous tenterons de répondre à notre problématique, à savoir le développement d'une méthode de modélisation à haut niveau d'abstraction dans un environnement unifié donnant les fonctionnalités d'un système hétérogène et intégrant les principaux phénomènes d'intégrité du signal, à l'aide du langage SystemC-AMS et d'un réseau de neurones.

Chapitre 3

Modélisation

Sommaire

3.1	Introduction	35
3.2	Présentation des outils	37
3.2.1	SystemC-AMS	37
3.2.1.1	Présentation des trois modèles de calcul	38
3.2.1.2	Synthèse et choix du MoC le plus adapté	44
3.2.2	Réseaux de neurones artificiels	44
3.2.2.1	Introduction	44
3.2.2.2	Neurones artificiels	46
3.2.2.3	Architecture des réseaux	47
3.2.2.4	Choix de l'architecture et des paramètres associés	49
3.2.2.5	Apprentissage	51
3.2.2.6	Choix du type d'apprentissage	52
3.2.3	Conclusion	52
3.3	Blocs fonctionnels	52
3.4	Blocs non-fonctionnels	53
3.4.1	Introduction	53
3.4.2	Etape 1 - Acquisition des couples d'entrées/sorties-désirées	54
3.4.3	Etape 2 - Apprentissage du réseau de neurones	55
3.4.4	Etape 3 - Construction du modèle SystemC-AMS TDF implémentant le réseau de neurones et les adaptateurs	57
3.4.4.1	Implémentation du réseau de neurones	59
3.4.4.2	Adaptateurs	60
3.4.5	Conclusion	61
3.5	Conclusion	61

3.1 Introduction

Dans le chapitre précédent, nous avons, d'une part, retenu le langage SystemC-AMS pour modéliser les fonctionnalités d'un système hétérogène à haut niveau d'abstraction, et d'autre part, les réseaux de neurones pour modéliser les phénomènes d'intégrité du signal. Dans ce chapitre, nous allons présenter notre méthodologie à l'aide de ces deux outils pour

répondre à notre problématique, à savoir l'apport d'une méthode de modélisation dans un environnement unifié :

1. donnant les principaux phénomènes d'intégrité du signal à haut niveau d'abstraction
2. et les intégrant avec les fonctionnalités d'un système hétérogène.

Notre méthodologie repose sur la combinaison de deux types de modèles complémentaires, nommés blocs fonctionnels et blocs non-fonctionnels.

Les blocs fonctionnels servent à modéliser le comportement idéal du système. Par exemple, pour le système présenté dans la figure 1.3, les fonctions de tous les nœuds (capteur, moteur, RAM, DSP, etc.), les contrôleurs d'E/S et même les fonctions du bus seront modélisées dans ces blocs. En considérant que les messages sont transmis sous forme numérique sur le bus de communication, les messages échangés entre les nœuds sont donc représentés par des valeurs binaires. Ces blocs sont décrits en langage C/C++ pour la partie logicielle, et en SystemC/SystemC-AMS pour la partie matérielle.

Les blocs non-fonctionnels servent eux à modéliser le comportement réel des signaux générés par le système et toutes leurs non-idéalités. Pour représenter ces phénomènes d'intégrité du signal sur les lignes de transmission ou les ports d'E/S, les signaux sont exprimés sous forme analogique. Un réseau de neurones est utilisé pour modéliser ces signaux à haut niveau d'abstraction. Ce réseau de neurones est construit dans un modèle TDF (*Timed Data Flow*) de SystemC-AMS pour faciliter son intégration et son interaction avec les blocs fonctionnels. Enfin, pour faire l'interface entre la représentation logique des données dans les blocs fonctionnels et la représentation réelle dans les blocs non-fonctionnels, on ajoute au modèle du système des adaptateurs qui effectuent la conversion réelle/logique ou logique/réelle.

La figure 3.1 résume notre méthodologie de modélisation pour un exemple de système (voir la figure 1.3). Dans la figure 3.1.a, le modèle composé uniquement des blocs fonctionnels permet de simuler de façon autonome les fonctionnalités du système. La figure 3.1.b ajoute les blocs non-fonctionnels pour prendre en compte les performances d'intégrité du signal.

Dans ce chapitre, nous présentons tout d'abord les deux outils nécessaires à notre méthode (section 3.2). Puis, dans la section 3.3, nous présentons brièvement les blocs fonctionnels. Dans la section 3.4, nous détaillons la constitution de nos blocs non-fonctionnels. Enfin, nous concluons ce chapitre par un bilan de notre méthodologie, où nous ferons apparaître ses avantages potentiels.

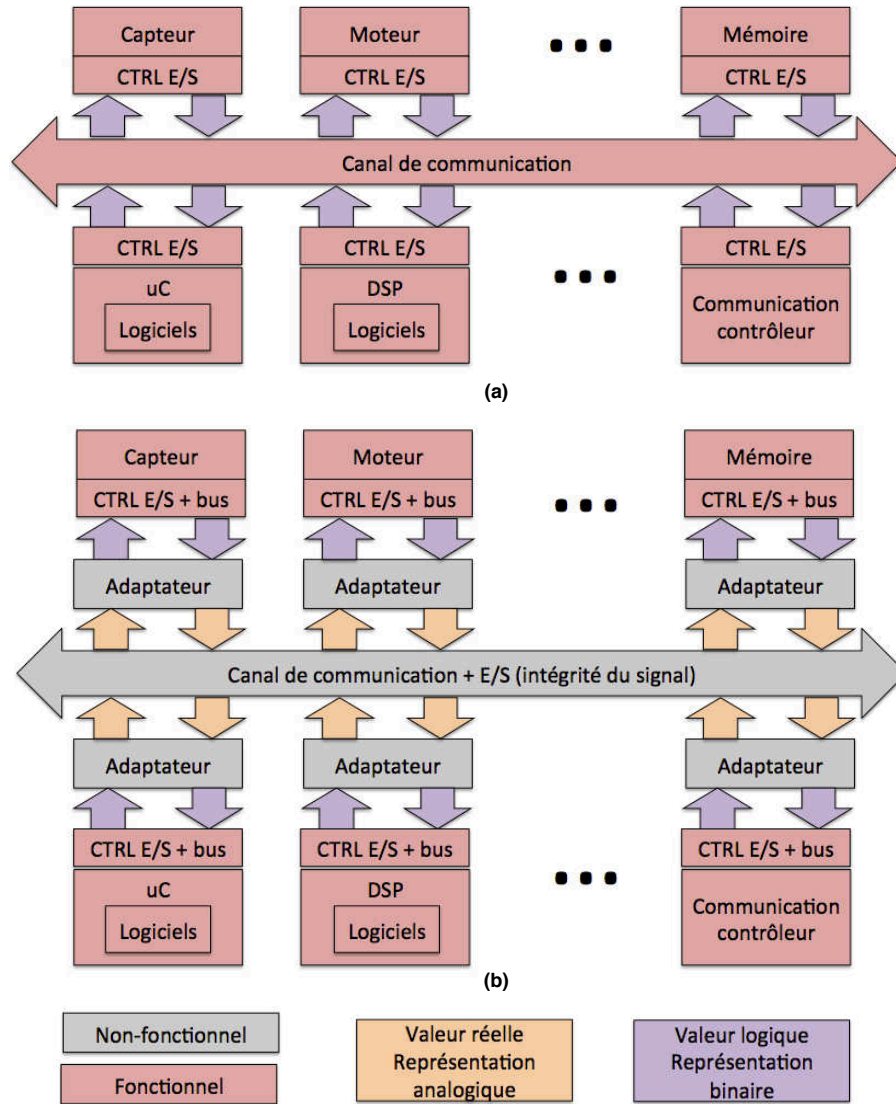


FIGURE 3.1 – Schema de principe de notre méthodologie : (a) Blocs fonctionnels seuls et (b) Intégration des blocs non-fonctionnels avec les blocs fonctionnels.

3.2 Présentation des outils

3.2.1 SystemC-AMS

Comme nous l'avons montré brièvement dans le chapitre 2, SystemC-AMS est une extension de SystemC pour la modélisation de composants analogiques ou mixtes. Elle ajoute trois nouveaux modèles de calcul (*Model of computation*, MoC) au noyau SystemC. Ces trois MoCs travaillent à des niveaux d'abstraction différents. Pour modéliser un système mixte complexe, il est possible de les associer. La figure 3.2 présente les trois MoCs et leurs principales caractéristiques (simulation à temps discret ou continu, respect ou non des lois de Kirchhoff). La figure 3.3 présente les types de signaux manipulés par chaque MoC. On voit que TDF (*Timed Data Flow*) fonctionne à temps discret. Comme LSF (*Linear Signal Flow*), il travaille avec des valeurs numériques sans qu'il y ait nécessairement de

lien avec des grandeurs électriques, alors que ELN (*Electrical Linear Network*) manipule des courants ou des tensions et respecte donc les lois de Kirchhoff.

Type de simulation	Temps discret Non-linéaire	Temps continu Linéaire	
		Non-respect des lois de Kirchhoff (Non-conservatif)	Respect des lois de Kirchhoff (Conservatif)
Modèle de calcul	Timed Data Flow (TDF)	Linear Signal Flow (LSF)	Electrical Linear Networks (ELN)

FIGURE 3.2 – Caractéristiques de MoCs de SystemC-AMS.

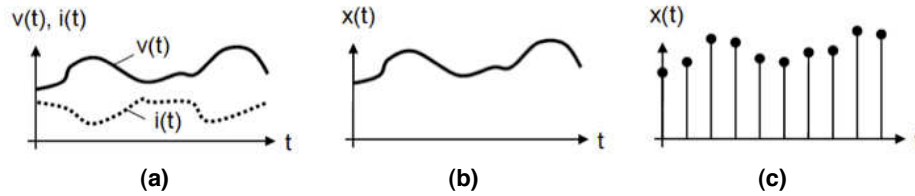


FIGURE 3.3 – Forme des signaux dans les MoCs : (a) ELN, (b) LSF et (c) TDF

3.2.1.1 Présentation des trois modèles de calcul

i. TDF : Timed Data Flow

Le MoC TDF est basé sur un autre formalisme de modélisation bien connu, le SDF (*Synchronous Data Flow*). Contrairement au modèle SDF non temporisé, la sémantique d'exécution de TDF introduit une modélisation à temps discret. Cette caractéristique va permettre d'établir au début de la simulation un ordonnancement statique qui définit l'ordre d'exécution des blocs du modèle TDF, et ainsi optimiser l'exécution de ces modèles en fonction des flux de données générés, pour atteindre des vitesses de simulation élevées.

Un modèle (ou cluster) TDF est typiquement composé d'un ensemble de modules. Chaque module comprend une méthode *processing()* dans laquelle on donne le comportement du module. Ce comportement peut être décrit d'une façon abstraite ou idéale ou bien de façon plus précise. Par exemple, le code de la figure 3.4 donne deux descriptions possibles d'un amplificateur. Dans la figure 3.4.a, l'amplificateur est modélisé par une formule idéale. Dans la figure 3.4.b, la description est raffinée en considérant les phénomènes de saturation. Un module TDF dispose également de méthodes optionnelles, comme *set_attributes()* qui permet de définir les caractéristiques du module (ou d'un port).

Les modules dans un cluster sont connectés par des ports qui sont unidirectionnels. La figure 3.5 présente un cluster TDF de trois modules. Un module TDF peut avoir plusieurs ports d'entrée et de sortie. Par exemple, dans la figure 3.5, le module A contient un port de sortie, le module B contient un port d'entrée et un port de sortie, le module C contient un

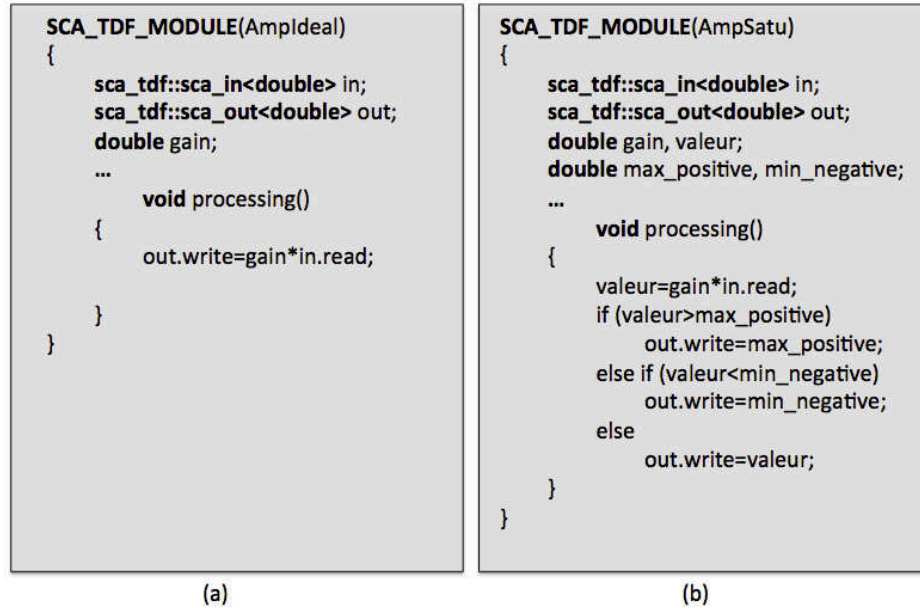


FIGURE 3.4 – Deux descriptions différentes d’un amplificateur en SystemC-AMS TDF.

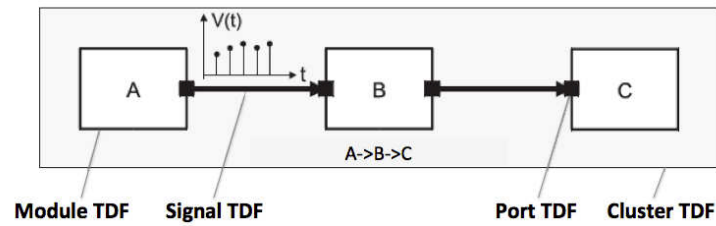


FIGURE 3.5 – Structure d’un cluster TDF.

port d’entrée. Puis, la fonction f (ex. f_A , f_B et f_C) réalisée dans la méthode *processing()* de chaque module ne dépend que de ses ports d’entrée ou de ses états internes. Ainsi, la fonction globale présentée par le cluster de la figure 3.5 est $f_C(f_B(f_A(...)))$.

Dans la méthode *set_attributes()*, on peut également attribuer aux ports TDF trois paramètres importants :

- Le pas de temps : il définit l’intervalle entre deux échantillons. Par exemple, si le pas de temps est fixé à 1 ns, le module sera activé à chaque nanoseconde.
- Le délai : il définit le retard entre l’acquisition et l’utilisation d’un échantillon. Par exemple, si le délai d’un port d’entrée est fixé à n , l’échantillon présenté à l’instant T ne sera pris en compte qu’à l’instant $T + n$.
- Le débit d’échantillonnage : il définit le nombre de valeur lues ou écrites à chaque activation. Par exemple, si le débit d’échantillonnage est fixé à n , le port devra lire ou écrire n valeurs à chaque activation.

Il est nécessaire au minimum de spécifier un pas de temps sur un des ports de l’un des modules du système. Les valeurs de ces différents paramètres sont vérifiées pour l’ensemble des modules au début de la simulation, ce pour des raisons de cohérence. En cas d’inco-

hérence, il n'est pas possible d'établir un ordonnancement pour l'activation des modules, et la simulation est interrompue. Par exemple, dans la figure 3.6.a, le pas de temps du module A est $T = 20 \mu s$. Son débit d'échantillonnage est $R = 2$. Alors, à chaque activation du module A (toutes les $20 \mu s$), la sortie de ce module produit deux échantillons. Les pas de temps des modules B et C sont fixés à $T = 10 \mu s$ et leur débit d'échantillonnage est $R = 1$. Alors, l'ordonnancement de ce cluster TDF est $A \rightarrow B \rightarrow C \rightarrow B \rightarrow C$. Si on modifie le pas de temps du module A à $T = 10 \mu s$ comme le montre la figure 3.6.b. Toutes les $10 \mu s$, l'un des deux échantillons du module A est perdu par le module B. L'ordonnancement ne peut pas être établi, ce qui conduit une erreur de simulation.

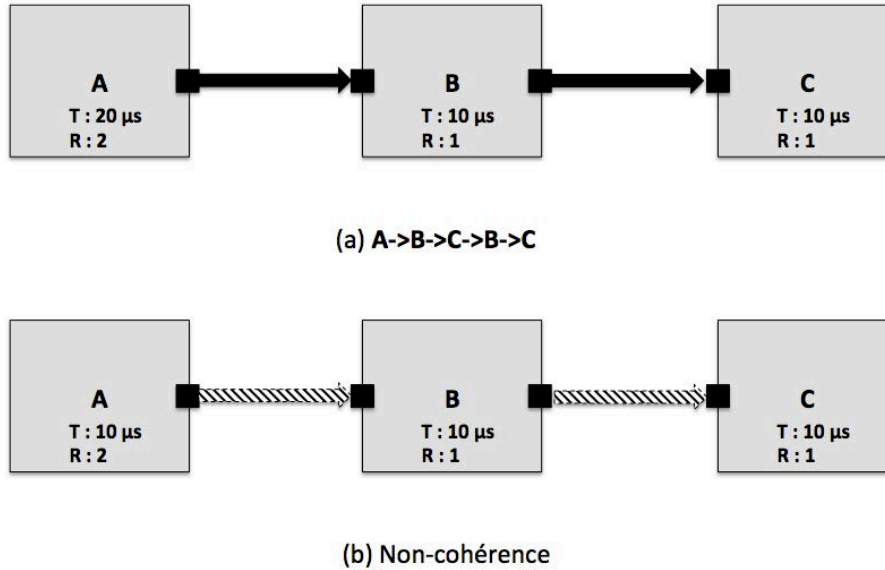


FIGURE 3.6 – Exemples d'ordonnancements (a) cohérent et (b) incohérent.

Le moteur de simulation TDF peut fonctionner de 2 façons différentes :

- TDF conventionnel : dans la version 1.0 [sys10] de systemC-AMS, les paramètres sont fixés de manière statique au début de la simulation et ne peuvent plus être modifiés par la suite pour des raisons de simplicité et d'efficacité. Dans le module *set_attributes()*, les valeurs de ces paramètres doivent être fixées par la fonction *set_timestep()*.
- TDF dynamique : l'objectif de TDF dynamique est d'offrir des mécanismes pour faire varier dynamiquement certaines propriétés d'un modèle TDF, tels que son pas de temps, le débit des échantillons ou le délai. Dans [BEGTM11], les auteurs ont indiqué les limitations de TDF conventionnel. Le TDF conventionnel avec son pas de temps fixe ne permet pas de modéliser efficacement des systèmes ou des comportements permettant de varier en fréquence au cours de la simulation (comme un VCO, une PWM, du jitter, ...). Alors, pour surmonter efficacement ces limites, les auteurs ont introduit le TDF dynamique, qui est implémenté dans la version 2.0 [sys12] de SystemC-AMS.

Pour illustrer les avantages du TDF dynamique, nous présentons un modèle qui réalise une modulation de fréquence. Cette fonction est présentée dans la figure 3.7. Quand le signal de contrôle est maintenu à "1", la sortie OUT est une sinusoïde d'une fréquence de 1,000 Hz. Si le signal de contrôle reste à "0", la sortie OUT est une sinusoïde d'une fréquence

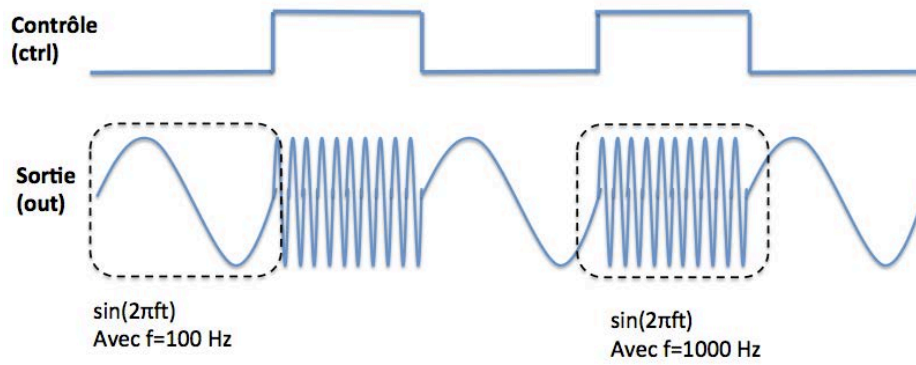


FIGURE 3.7 – Illustration d'une modulation de fréquence.

de 100 Hz. Dans la figure 3.8, ce bloc est modélisé en TDF conventionnel. Sachant que la fréquence maximale de la sortie est de 1,000 Hz, on fixe le pas de temps à 0.01 ms dans la méthode `set_attributes()`. Le module TDF va donc être activé 1,000 fois par période si $f(OUT) = 100$ Hz, et 100 fois par période si $f(OUT) = 1,000$ Hz. La figure 3.9 montre l'implémentation de cette même fonction en TDF dynamique. Ce moteur de simulation fournit une méthode `change_attributes` pour décider des conditions du changement de la valeur du pas de temps, et ajoute des nouvelles fonctions dans la méthode `set_attributes()` pour valider ces modifications. Dans notre cas, nous allons adapter la valeur du pas de temps pour que, quelle que soit la fréquence du signal OUT, on n'active le module TDF que 100 fois par période. Ainsi, si $f(OUT) = 1,000$ Hz (ctrl="1"), le pas de temps est fixé à 0.01 ms. Si $f(OUT) = 100$ Hz (ctrl="0"), le pas de temps est fixé à 0.1 ms. On s'assure ainsi d'un meilleur compromis entre précision et vitesse de simulation. La comparaison des deux modules est donnée dans le tableau 3.1. On peut constater que si la fréquence du signal de sortie est à 100 Hz, le module TDF conventionnel sera activé 10 fois plus souvent que le module TDF dynamique. La précision obtenue avec le TDF conventionnel est certes meilleure que pour le TDF dynamique, mais cela se fait au détriment de la vitesse de simulation. Si le concepteur n'a pas besoin d'un tel degré de précision, le modèle TDF dynamique est alors globalement plus efficace que le modèle TDF conventionnel.

TABLE 3.1 – Comparaison entre le module TDF conventionnel et le module TDF dynamique.

Type de TDF	Pas de temps (ms)		Activations par période de OUT	
	f=100 Hz	f=1,000 Hz	f=100 Hz	f=1,000 Hz
TDF conventionnel	0.01	0.01	1,000	100
TDF dynamique	0.1	0.01	100	100

ii. LSF : Linear Signal Flow

Le MoC LSF permet de modéliser le comportement d'un système sous la forme d'un ensemble d'équations algébriques linéaires (*Differential Algebraic Equation*, DAE). LSF

```

SCA_TDF_MODULE(FM)
{
    sca_tdf::sca_in<double> ctrl;
    sca_tdf::sca_out<double> out;

    double freq=100; // Fréquence
    sca_core::sca_time t_step=sca_core::sca_time(0.01,sc_core::SC_MS); // Pas de temps

    void set_attributes()
    {
        set_timestep( t_step ); // Définir le pas de temps = 0.01 ms
    }

    void processing()
    {
        double t=get.time().to_second; // Mesure du temps de la simulation
        if (ctrl.read()==1) // Mise à jour du signal de contrôle
        {
            out.write(sin(2.0*M_PI*10*freq*t)); // out=sin(2πft) avec f=1000 Hz
        }
        else
        {
            out.write(sin(2.0*M_PI*freq*t)); // out=sin(2πft) avec f=100 Hz
        }
    }
    ...
}

```

FIGURE 3.8 – Module TDF conventionnel pour la modulation de fréquence.

```

SCA_TDF_MODULE(FM)
{
    sca_tdf::sca_in<double> ctrl;
    sca_tdf::sca_out<double> out;
    sca_core::sca_time t_step;

    void set_attributes()
    {
        set_timestep( t_step );
        does_attribute_changes(); // Accepter le changement des attributs du module TDF
        accept_attribute_changes(); // Accepter le changement des attributs du module TDF
    }

    void processing()
    { ... // Identique à la figure 3-8 }

    void change_attributes()
    {
        if (ctrl.read()==1)
        {
            t_step=sca_core::sca_time(0.01,sc_core::SC_MS); // Pas de temps = 0.01 ms
        }
        else
        {
            t_step=sca_core::sca_time(0.1,sc_core::SC_MS); // Pas de temps = 0.1 ms
        }
        request_next_activation( t_step ); // Attendre la prochaine activation du module
    }
    ...
}

```

FIGURE 3.9 – Module TDF dynamique pour la modulation de fréquence.

est un formalisme de modélisation en temps continu. Il décrit le système sous la forme d'association de blocs fournis dans la bibliothèque. Chaque bloc représente une fonction élémentaire, tels que additionneur, multiplieur, intégrateur, bloc de retard, fonction de transfert ou représentation en états. L'ensemble de ces blocs définit un système d'équations linéaires qui est résolu par un solveur linéaire. Par conséquent, seules les fonctions mathématiques linéaires sont disponibles.

Le formalisme LSF est particulièrement adapté pour la description comportementale des systèmes asservis linéaires. La connexion de ces blocs élémentaires se fait par des ports d'entrées/sorties qui sont unidirectionnels, comme les ports TDF. La figure 3.10 montre un exemple d'un modèle LSF.

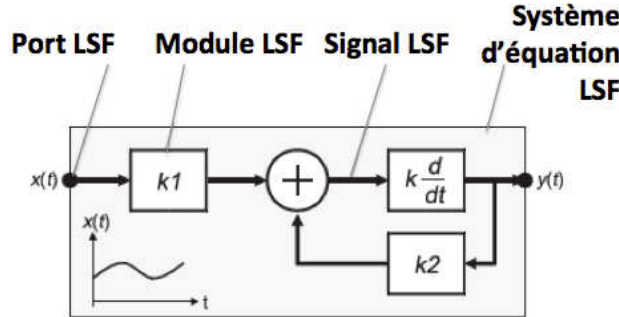


FIGURE 3.10 – Structure d'un modèle LSF.

Pour connecter un modèle LSF avec un modèle TDF, SystemC-AMS propose un convertisseur TDF vers LSF (`sca_lsf : sca_tdf : sca_source`) et un convertisseur LSF vers TDF (`sca_lsf : sca_tdf : sca_sink`).

iii. ELN : Electrical Linear Networks

Le MoC ELN repose sur l'utilisation de composants électriques linéaires fondamentaux tels que des résistances ou des condensateurs et de leurs interconnexions électriques pour modéliser les comportements conservatifs des lois de Kirchhoff en temps continu. Ces composants électriques linéaires peuvent être reliés ensemble, afin de former un réseau électrique. La figure 3.11 montre un exemple d'un réseau électrique modélisé avec le MoC ELN. ELN définit deux types de connexions : nœud et terminal. Le nœud permet de connecter les composants entre eux. Le terminal permet de connecter les différents réseaux ELN entre eux. Contrairement aux ports TDF et LSF, les nœuds et les terminaux ELN sont bidirectionnels en fonction du sens de propagation du courant.

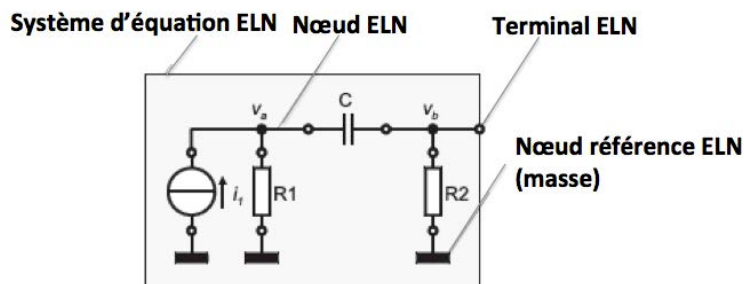


FIGURE 3.11 – Structure d'un modèle ELN.

Dans un réseau ELN, les relations mathématiques entre les primitives électriques sont définies en chaque nœud du réseau. La tension et le courant sont définis par les lois de Kirchhoff. En tant que tel, le réseau électrique est représenté par un ensemble d'équations algébriques différentielles, qui sont résolues pendant la simulation pour déterminer le comportement du circuit. Par conséquent, seuls des composants au comportement linéaire

sont inclus dans la liste des primitives. Ainsi, il n'y a pas de transistors ou de diode.

Par ailleurs, pour connecter un module ELN avec un module TDF, SystemC-AMS propose des convertisseurs TDF vers ELN et ELN vers TDF :

- Pour la tension : `sca_eln::sca_tdf::sca_vsource, sca_eln::sca_tdf::sca_vsink`.
- Pour le courant : `sca_eln::sca_tdf::sca_isource, sca_eln::sca_tdf::sca_isink`.

Ces convertisseurs (dans le cas d'un échange ELN vers TDF) extraient la valeur de la tension ou du courant en un point du circuit et le transfèrent vers un signal TDF en sortie. La figure 3.12 schématise l'exemple d'une conversion d'une tension ELN en un signal TDF.

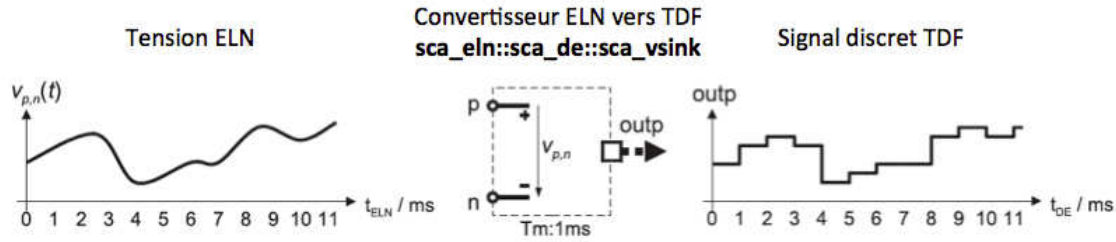


FIGURE 3.12 – Convertisseur de tension ELN vers TDF.

3.2.1.2 Synthèse et choix du MoC le plus adapté

Les caractéristiques principales de ces MoCs sont résumées dans le tableau 3.2. Comme nous l'avons dit, la conception des systèmes AMS nécessite la combinaison de différents MoC pour travailler à différents niveaux d'abstraction. Ceci requiert des moyens de communication ou de synchronisation entre les différents MoC. Cela est possible à l'aide des convertisseurs fournis par SystemC-AMS. Par ailleurs, les connections entre le domaine analogique (description en SystemC-AMS) et le domaine numérique (description en SystemC) se font grâce à des ports spécialisés qui vont échanger les données et surtout garantir la bonne synchronisation entre les deux domaines. Chaque MoC de SystemC-AMS possède ses propres convertisseurs vers le domaine numérique.

Parmi les trois MoCs, le TDF est le MoC de SystemC-AMS travaillant au niveau d'abstraction le plus élevé. Il permet en outre de modéliser les comportements non-linéaires. Par conséquent, il est le MoC le plus adapté à nos besoins.

3.2.2 Réseaux de neurones artificiels

3.2.2.1 Introduction

Un neurone biologique est une cellule excitable constituant l'unité fonctionnelle de base du système nerveux. Le nombre total de neurones biologiques du cerveau humain est estimé entre 86 et 100 milliards [GC90]. Comme le montre la figure 3.13, un neurone biologique est constitué de trois parties : l'axone, la dendrite et le noyau. Les noyaux des neurones reçoivent les signaux (impulsions électriques) par les dendrites et envoient l'information par les axones. Ces signaux traversent un espace vide qui se trouve entre la dendrite et l'axone, appelé synapse, par un transfert bio-chimique. Chaque neurone intègre

TABLE 3.2 – Bilan des MoCs de SystemC-AMS

	MoC		
	<i>TDF</i>	<i>LSF</i>	<i>ELN</i>
Représentation temporelle	Discret	Continu	Continu
Forme de données	Valeurs numériques échantillonnés	Signaux abstraits continus	Signaux électriques (tension et courant)
Type d'interface	Ports unidirectionnels	Ports unidirectionnels	Terminaux et nœuds bidirectionnels
Description comportementale	Algorithmes	DAE	DAE
Type de comportement	Linéarités et non-linéaires	Opérations mathématiques linéaires	Lois de Kirchhoff

jusqu'à un millier de signaux synaptiques. Ces signaux opèrent de manière non-linéaire. Quand la somme dépasse un certain seuil, le neurone émet un signal électrique. Suivant le type de la synapse, le signal électrique d'un neurone peut renforcer ou diminuer l'activité de ses voisins. On parle ainsi de synapse excitatrice ou inhibitrice.

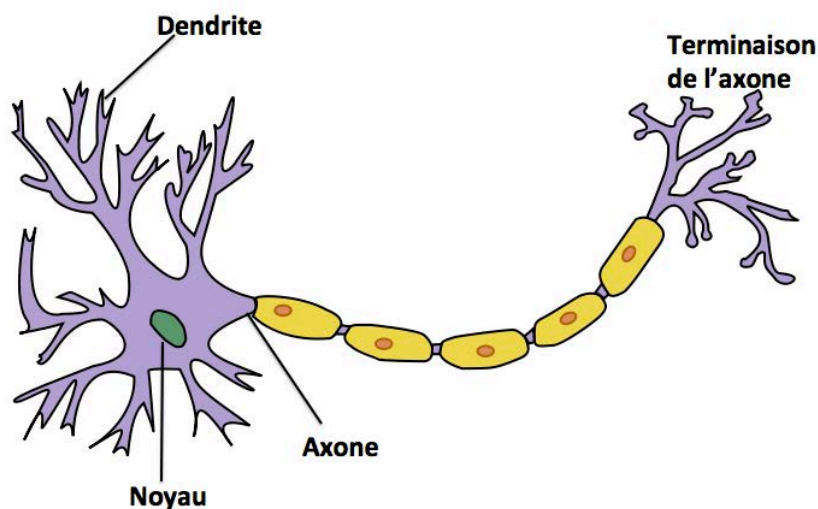


FIGURE 3.13 – Schéma de principe d'un neurone biologique [wik].

Un réseau de neurones artificiels est un modèle de calcul inspiré du fonctionnement des neurones biologiques. Dans la section suivante, nous en présentons les principes.

3.2.2.2 Neurones artificiels

Comme le montre la figure 3.14.a, la structure d'un neurone artificiel est inspirée de la structure des neurones biologiques. Sur cette figure, $X_0, \dots, X_n$ sont les données d'entrée du neurone et représentent les dendrites du neurone biologique. $W_0, \dots, W_n$ sont les poids des entrées et représentent les synapses. Ces poids peuvent être positifs comme les synapses excitatrices, ou négatifs comme les synapses inhibitrices. Le bias b_k est une polarisation qui augmente ou diminue la somme des entrées $X_i W_i$. La somme pondérée des entrées du neurone est notée S_k et est donnée par l'équation 3.1 :

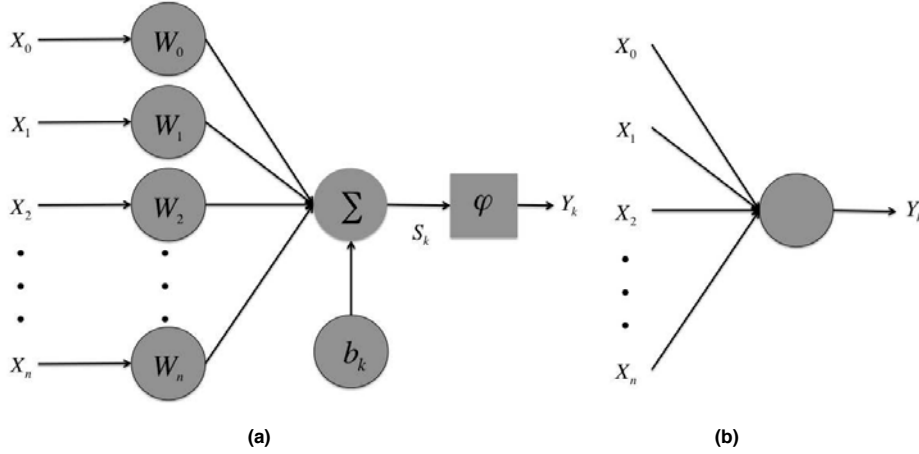


FIGURE 3.14 – a) Exemple de neurone artificiel, b) Symbole simplifié.

$$S_k = b_k + \sum_{i=1}^n (X_i \cdot W_i). \quad (3.1)$$

Y_k est la sortie du neurone et représente l'axone du neurone biologique. La fonction φ est la fonction d'activation de transfert de S_k en Y_k . Cette fonction est souvent non linéaire et peut être attribuée par l'utilisateur, selon le rôle du neurone. Les fonctions d'activation présentées dans la figure 3.15 sont fréquemment utilisées [Hay99]. La sortie d'un neurone est donc exprimée par l'équation 3.2 :

$$Y_k = \varphi(S_k). \quad (3.2)$$

Pour plus de clarté dans le cas de réseaux complexes, le neurone de la figure 3.14.a peut être représenté plus simplement sous la forme de la figure 3.14.b. Dans la suite de cette thèse, cette représentation sera souvent reprise.

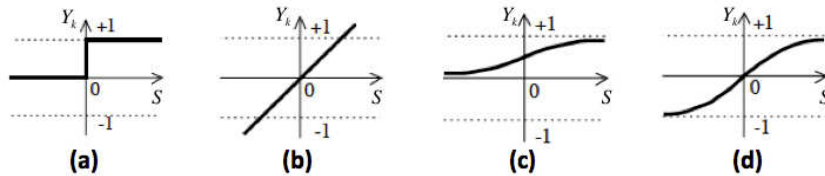


FIGURE 3.15 – Exemples de fonctions d'activation (a) hard-limit, (b) identité, (c) log-sigmoïde, et (d) tan-sigmoïde.

Un réseau de neurones est, comme son nom l'indique, un ensemble de neurones interconnectés, où les sorties des neurones deviennent des entrées d'autres neurones. L'architecture générale des réseaux organise les neurones en couches successives. La première couche est la couche d'entrée. Cette couche ne contient pas vraiment de neurones traitants [Hay99, HDB96]. Elle permet de normaliser les valeurs des entrées, puis de les distribuer vers la couche suivante. La dernière couche est la couche de sortie. Les couches intermédiaires sont les couches cachées du réseau de neurones. Ces couches sont appelées cachées car elles n'ont aucun contact avec l'extérieur du réseau. Pour bâtir un réseau de neurones, il faut tout d'abord choisir son architecture, par exemple, monocouche vs multicouches, non-bouclé vs bouclé, les fonctions d'activation, le nombre de neurones dans la couche cachée. Ces différentes architectures ont des propriétés différentes. Par exemple, pour un problème linéaire, un réseau de neurones monocouche est suffisant. Mais, pour un problème non linéaire, un réseau multicouches est en général plus adapté.

Les trois structures d'architecture de réseau sont présentées dans la section 3.2.2.3. Nous présentons ensuite comment choisir une architecture dans la section 3.2.2.4. Après le choix d'architecture du réseau, une procédure d'apprentissage est nécessaire. Le but de cette procédure est de trouver les bonnes valeurs des poids des connexions (W_i) et des bias (b_i), pour aussi fournir les sorties désirées. Les différents algorithmes d'apprentissage sont présentés dans la section 3.2.2.5 et le choix d'apprentissage pour notre thèse est donné dans la section 3.2.2.6.

Les réseaux de neurones sont utilisés dans de très nombreux domaines [HDB96], comme la classification, la modélisation, l'analyse, etc. Par exemple, dans [LBBH98], un réseau de neurones est employé pour la reconnaissance de caractères manuscrits. Dans [GD04], c'est pour la détection et la reconnaissance de visages. Dans le domaine électronique, on peut citer [VNZ97], où un réseau de neurones est utilisé pour simuler le champ électromagnétique et pour optimiser les interconnexions d'un circuit VLSI ; dans [FYWZ00], un réseau de neurones bouclé est créé pour apprendre les réponses dynamiques d'un circuit non-linéaire ; dans [CEZ10], les auteurs proposent de modéliser les interfaces non-linéaires par une méthode combinant un réseau de neurones et un circuit équivalent.

3.2.2.3 Architecture des réseaux

Trois différents types d'architectures de réseaux de neurones ont été définis dans [Hay99].

Réseaux de neurones monocouches non-bouclés :

Comme le montre la figure 3.16, ce type de réseaux de neurones ne contient qu'une couche d'entrée et une couche de sortie. Les caractéristiques de ce type de réseaux ne diffèrent pas de celles d'un neurone unique et ils peuvent être utilisés pour résoudre des problèmes relativement simples comme de la classification, ou bien résoudre des opérations logiques simples (ET, OU, ...). Cette structure est peu efficace pour aborder des problèmes non linéaires.

Réseaux de neurones multicouches non-bouclés :

Pour des applications comportant des non-linéarités, il faut enrichir la structure du réseau monocouche en ajoutant une ou plusieurs couches cachées entre la couche d'entrée et la couche de sortie.

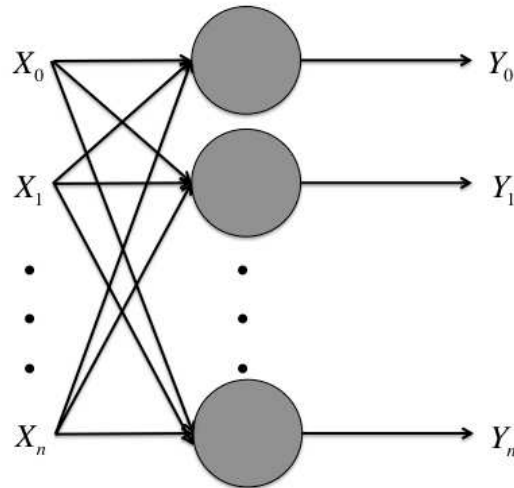


FIGURE 3.16 – Schéma de principe d'un réseau monocouche.

La figure 3.17 montre un réseau multicouches non bouclé. Il comporte n entrées, un certain nombre de couches cachées, et une couche de sortie. Le rôle des couches cachées est d'établir une relation entre les entrées et la couche de sortie par une fonction choisie. Le sens de transfert de l'information dans un réseau non bouclé est défini par l'ordre des couches. Il est toujours dirigé de la couche d'entrée vers la couche de sortie.

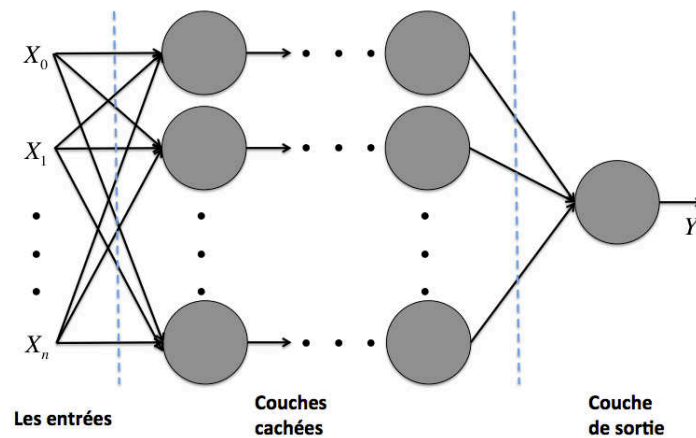


FIGURE 3.17 – Schéma de principe d'un réseau multicouches non-bouclé.

On dit que le réseau est entièrement connecté si dans chaque couche du réseau, chaque nœud est connecté à l'ensemble des nœuds de la couche suivante (figure 3.18.a). Si cette condition n'est pas remplie, on dit que le réseau est partiellement connecté (figure 3.18.b).

Réseaux de neurones bouclés :

Le spécificité d'un réseau de neurones bouclé est de comporter au moins une boucle de rétroaction. Par exemple, la couche de sortie du réseau de neurones présenté dans la figure 3.19 propage un signal de vers l'entrée.

La présence de boucles de rétroaction a un impact profond sur les caractéristiques et les performances du réseau. Une boucle de rétroaction implique nécessairement l'utilisation

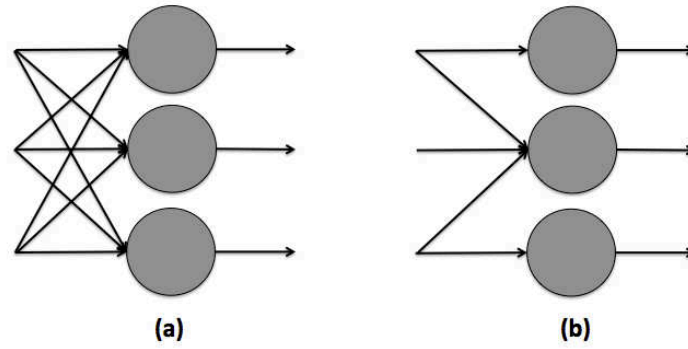


FIGURE 3.18 – Deux types de connexions : (a) entière et (b) partielle.

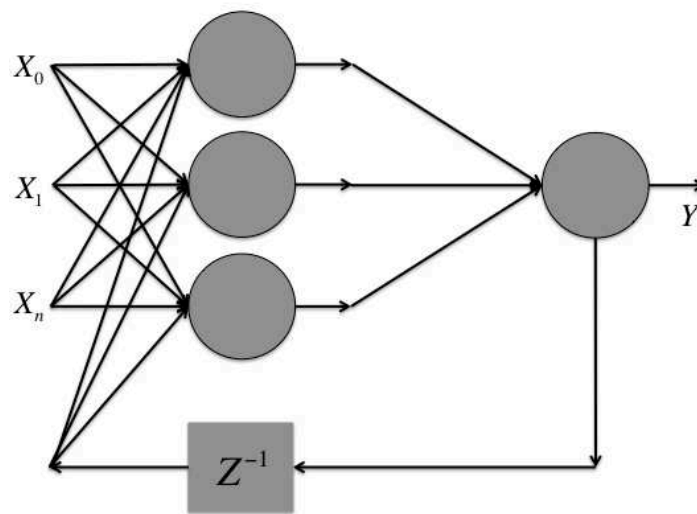


FIGURE 3.19 – Exemple de réseau de neurones bouclé.

d'unités de retard (notées Z^{-1}). La sortie d'une unité de retard est décalée d'une unité de temps par rapport à son entrée. Le nombre d'unités de retard à insérer dépend des applications du réseau. Dans le cas où l'on a inséré n unités de retard, on a alors l'équation 3.3 :

$$Z^{-n}(y(t)) = y(t - n). \quad (3.3)$$

Dans le cas particulier de la figure 3.19, on a l'équation 3.4 :

$$Z^{-1}(y(t)) = y(t - 1). \quad (3.4)$$

A l'aide de ces éléments, le réseau peut représenter des comportements dynamiques non linéaires.

3.2.2.4 Choix de l'architecture et des paramètres associés

Selon les différents cas d'étude, les utilisateurs ont à choisir l'une de ces trois structures d'architecture, ainsi que ses paramètres (choix du nombre de couches, nombre de neurones dans la couche cachée, etc.). Un réseau de neurones à trois couches peut approximer la

plupart des fonctions [HDB96]. Aussi, les réseaux utilisés dans notre thèse sont constitués d'une ou de plusieurs entrées, d'une couche cachée et d'une couche de sortie. Le choix d'un réseau non-bouclé ou bouclé dépend du système ou des phénomènes à modéliser. Par exemple, pour modéliser un délai de propagation, un réseau non-bouclé est suffisant mais il faut ajouter des délais pour la couche d'entrée. Pour modéliser la diaphonie entre deux pistes adjacents, il faut un réseau bouclé (cf. figure 2.8). Si un réseau bouclé est choisi, il faut définir le nombre de délais. Généralement, ce nombre est déterminé de façon empirique.

Il faut également définir les autres caractéristiques de l'architecture choisie. Nous commençons par le choix des fonctions d'activation. Les fonctions d'activation sont les formules mathématiques qui déterminent les compétences du réseau, notamment la fonction d'activation de la couche cachée. Pour représenter des phénomènes non-linéaires de l'intégrité du signal, notre couche cachée doit utiliser une fonction d'activation non-linéaire, comme le log-sigmoïde ou le tan-sigmoïde. Le log-sigmoïde est typiquement utilisé pour apprendre la valeur moyenne d'un phénomène, alors que le tan-sigmoïde est utilisé pour apprendre la dérivation de la valeur moyenne [TT92]. Par ailleurs, l'apprentissage d'un réseau avec une fonction antisymétrique comme le tan-sigmoïde est plus rapide qu'avec une fonction non-symétrique (log-sigmoïde) [Hay99]. Nous choisissons donc pour nos réseaux le tan-sigmoïde (défini par l'équation 3.5) dans la couche cachée. Par contre, le choix d'une fonction linéaire dans la couche de sortie est motivé par le désir d'attirer l'attention sur la couche cachée pendant l'apprentissage [Hay99]. Dans notre thèse, la fonction purelin définie par l'équation 3.6 est donc utilisée dans la couche de sortie.

$$\text{tansig}(x) = \frac{2}{1 + e^{-2x}} - 1 \quad (3.5)$$

$$\text{purelin}(x) = x \quad (3.6)$$

Ensuite, il faut fixer le nombre de neurones dans la couche cachée. Ce nombre va définir les performances de la couche cachée. Si le nombre de neurones ne suffit pas à apprendre le problème correctement, on dit que le réseau de neurones a un problème de sous-apprentissage (*Underfitting*). Par contre, si le nombre de neurones est trop grand, il y a possibilité de conduire au problème de sur-apprentissage (*Overfitting*), le réseau perd alors de sa flexibilité, et saura donc moins bien réagir lorsque il sera confronté à des situations différentes de celles rencontrées lors de son apprentissage. Par ailleurs, un grand nombre de neurones prolonge le temps d'apprentissage. Pour déterminer le nombre de neurones optimal, il y a plusieurs méthodes heuristiques proposées, que nous recensons dans le tableau 3.3. Les méthodes données dans ce tableau considèrent seulement le nombre d'entrées ou de sorties. En effet, le nombre de neurones est défini par plusieurs paramètres, incluant le nombre d'entrées/sorties, la complexité du système à approximer, le nombre de données nécessaires à l'apprentissage du réseau et le type d'architecture du réseau de neurones. Il n'y a pas de méthode qui soit optimale pour l'ensemble des situations. Aussi, le nombre de neurones est plutôt déterminé de façon empirique.

Après le choix d'architecture du réseau et de ses paramètres, il faut encore faire son apprentissage pour trouver les bonnes connexions entre les neurones. Dans la section suivante, nous présentons les différentes méthodes d'apprentissage.

TABLE 3.3 – Heuristiques de calcul du nombre de neurones (N) dans la couche cachée (N_i : nombre d'entrées, N_o : nombre de sorties.).

Equation heuristique	Référence
$N \leq 2 \times N_i + 1$	[Nie87]
$N > 3N_i$	[Hus89]
$N = (N_i + N_o) / 2$	[Rip93]
$N = \sqrt{N_i \times N_o}$	[Rip93]
$N = 2N_i$	[KW97]

3.2.2.5 Apprentissage

Dans un réseau de neurones artificiels, les liens entre les synapses sont représentés par les poids des connexions entre les nœuds. L'apprentissage peut être considéré comme un problème d'établissement des poids des connexions, afin de fournir les sorties désirées. L'apprentissage est réalisé par des algorithmes de calcul. Une fois l'apprentissage réalisé, les poids sont fixés.

Deux phases peuvent être envisagées dans une procédure d'apprentissage. Ces deux phases sont :

1. la phase d'apprentissage : Un jeu d'entrées est présenté à la couche d'entrée. Les poids sont alors ajustés par des algorithmes itératifs à partir des valeurs initiales fixées aléatoirement au début de l'apprentissage.
2. la phase de validation : Un jeu d'entrées différent du précédent est présenté. Cette phase évalue la qualité de l'apprentissage. Dans le cas où le réseau de neurones passe le test, le réseau de neurones peut ensuite être utilisé pour résoudre le problème. Dans le cas contraire, il faut refaire la phase d'apprentissage.

Dans ce qui suit, l'apprentissage est présenté en détail. Trois catégories d'algorithmes d'apprentissage peuvent être utilisées [HDB96].

Apprentissage supervisé :

Pour ce type d'apprentissage, les utilisateurs fournissent au réseau de neurones des couples d'entrées/sorties-désirées, comme $\{X_1, T_1\}$, $\{X_2, T_2\}$, ... $\{X_n, T_n\}$, où X_1 à X_n sont les données d'entrée du neurone, et T_1 à T_n sont les sorties désirées. Par exemple, pour un système électrique, X_1 à X_n peuvent être les valeurs échantillonnées de la tension d'entrée et T_1 à T_n sont les valeurs échantillonnées de la tension de sortie. A chaque fois qu'une entrée X_i est présentée au réseau de neurones, la sortie Y_i est comparée à la sortie désirée. La distance entre Y_i et T_i représente l'erreur du réseau. Le but de l'apprentissage est d'ajuster la matrice de poids W et la matrice des biais b afin de minimiser l'erreur. Il existe plusieurs algorithmes pour ce type d'apprentissage, par exemple, l'algorithme Levenberg-Marquardt [Lev44, Don63, Sca85], l'algorithme Hebb [Heb49], etc.

Apprentissage par renforcement :

L'apprentissage par renforcement est similaire à l'apprentissage supervisé, sauf que, au

lieu de fournir les sorties désirées pour chaque entrée du réseau de neurones, cet algorithme fournit un grade. Le grade est un indicateur de la performance du réseau de neurones. Ce type d'apprentissage permet de savoir si la réponse du réseau de neurones est correcte ou non, mais sans connaître a priori la bonne réponse.

Apprentissage non-supervisé :

Contrairement à l'apprentissage supervisé, on ne fournit pas les valeurs des sorties désirées dans ce type d'apprentissage. Cet apprentissage est basé sur des probabilités. La matrice de poids W et la matrice des biais b sont modifiées seulement en fonction des relations statistiques des entrées du réseau de neurones, afin de produire des sorties qui soient proches des réponses pour des valeurs d'entrée similaires.

3.2.2.6 Choix du type d'apprentissage

L'apprentissage par renforcement semble être la procédure la plus adaptée pour des applications de commande [BSA83, WS92] et l'apprentissage non-supervisé est plutôt utilisé pour de la classification [HDB96].

Dans notre thèse, le réseau de neurones est utilisé pour modéliser les performances de l'intégrité du signal d'un système électronique et pour générer les sorties analogiques des composants. En conséquence, l'apprentissage supervisé est le plus adapté pour notre application.

3.2.3 Conclusion

Dans cette section, nous avons présenté les deux outils que nous allons utiliser pour développer notre méthodologie. Les fonctionnalités du système seront modélisées dans un ou plusieurs blocs fonctionnels en SystemC/SystemC-AMS. Les performances d'intégrité du signal seront modélisées grâce à un ou plusieurs réseaux de neurones implémentés en SystemC-AMS avec le moteur TDF. L'apprentissage supervisé permettra de modéliser le circuit à partir des signaux d'entrées/sorties.

Dans les sections suivantes, nous détaillons notre méthodologie, à commencer par la construction des blocs fonctionnels.

3.3 Blocs fonctionnels

La méthode que nous proposons est bâtie sur des blocs non-fonctionnels et des blocs fonctionnels, comme l'illustre la figure 3.1. Dans cette section, nous présentons les principes de modélisation de notre bloc fonctionnel. Dans la section suivante, nous présentons le bloc non-fonctionnel.

Comme nous l'avons présenté dans la partie introduction (section 3.1), les blocs fonctionnels sont bâtis pour les comportements idéaux du système. Comme la montre la figure 3.1, les blocs fonctionnels modélisent les fonctions idéales de tous les nœuds (capteur, moteur, RAM, logiciel embarqué). Selon le domaine de ces nœuds (analogique, numérique ou autre), ces blocs fonctionnels sont décrits en langage C/C++ ainsi qu'en SystemC/SystemC-AMS.

Les blocs fonctionnels modélisent aussi les fonctions du contrôleur d'E/S qui définissent les formes des messages transmis entre les nœuds. En considérant que les données sont échangées par un bus de communication, les signaux du contrôleur d'E/S sont présentés sous forme numérique. Par ailleurs, les fonctions idéales du bus doivent être aussi modélisées dans ces blocs, comme la fonction *wired-and* du bus I²C (cf. section 4.2).

Les blocs fonctionnels permettent de modéliser les fonctions du système de façon indépendante. En les associant avec les blocs non-fonctionnels, un modèle complexe est établi qui permet d'analyser les performances de l'intégrité du signal du système.

3.4 Blocs non-fonctionnels

3.4.1 Introduction

Dans cette section, nous présentons les détails de la modélisation de nos blocs non-fonctionnels.

Le bloc non-fonctionnel modélise les performances d'intégrité du signal du système. Il repose sur l'implantation dans un modèle SystemC-AMS TDF d'un réseau de neurones, qui donne le comportement d'un système en fonction de ses propriétés d'intégrité du signal. Grâce au réseau de neurones, ce comportement va être spécifié à un haut niveau d'abstraction, et ne va nécessiter qu'un nombre limité de paramètres. En plus du réseau, ce bloc inclut également un ou plusieurs adaptateurs qui permettent l'échange d'information entre les blocs fonctionnels et les blocs non-fonctionnels.

La méthode de construction du bloc non-fonctionnel consiste en trois étapes. Ces trois étapes sont présentées dans la figure 3.20 :

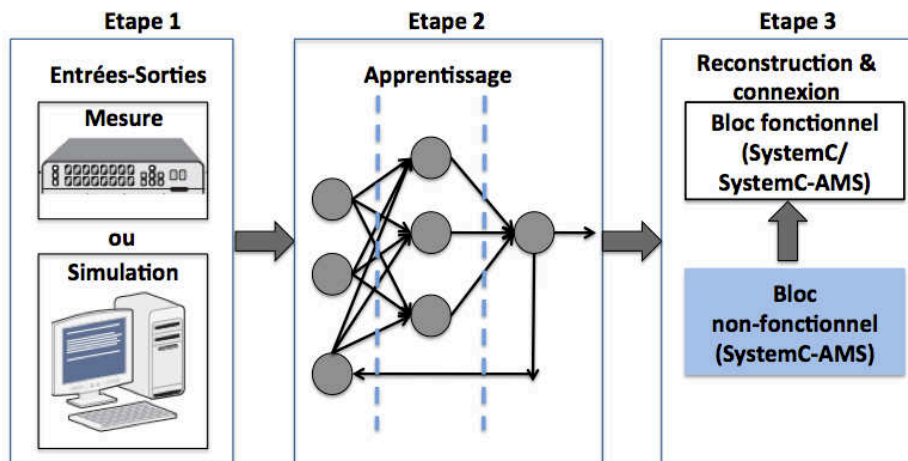


FIGURE 3.20 – Etapes nécessaires à la construction d'un bloc non-fonctionnel.

1. Acquisition des couples d'entrées/sorties-désirées du système à modéliser.
2. Apprentissage du réseau de neurones pour modéliser les performances d'intégrité du signal.
3. Implémentation du réseau de neurones en SystemC-AMS TDF et connexion avec le bloc fonctionnel à l'aide des adaptateurs.

Ces trois étapes sont détaillées dans les sections suivantes.

3.4.2 Etape 1 - Acquisition des couples d'entrées/sorties-désirées

Comme nous l'avons présenté dans la section précédente, nous avons choisi l'apprentissage supervisé parmi les trois types d'apprentissages possibles d'un réseau de neurones. Pour réaliser ce type d'apprentissage, il faut fournir les couples d'entrées/sorties-désirées au réseau de neurones. Le but de la première étape est d'acquérir ces couples d'entrées/sorties-désirées. La figure 3.21 présente cette méthode d'acquisition.

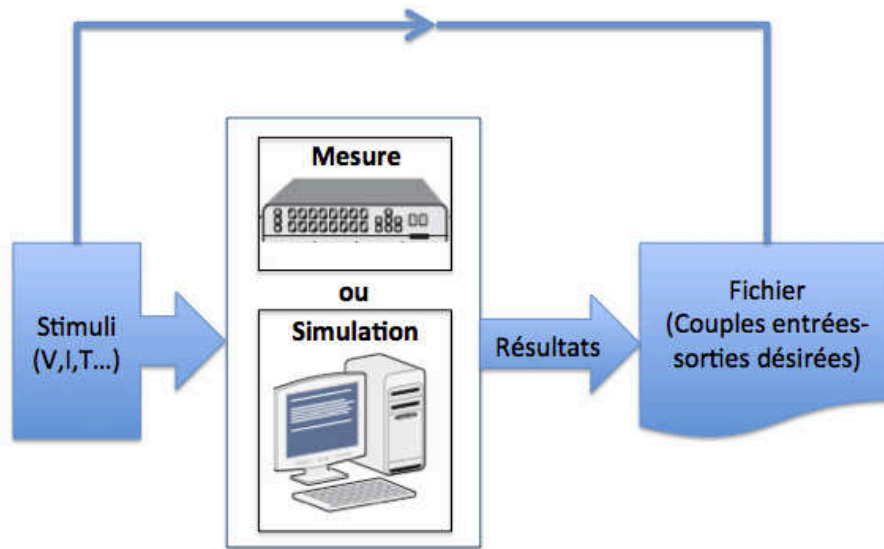


FIGURE 3.21 – Méthode d'acquisition des entrées/sorties.

Dans notre thèse, nous souhaitons caractériser les performances d'intégrité du signal d'un système électronique. Les couples d'entrées/sorties-désirées doivent donc nous donner les paramètres permettant d'établir ces performances. Dans notre cas, les paramètres pertinents sont les courants et les tensions en entrée ou en sortie des composants du système, mais également la température, ou encore les caractéristiques des lignes de transmission (comme la longueur de ces lignes). Selon le cas d'étude, d'autres paramètres peuvent être pertinents (voir chapitre suivant).

Nous pouvons acquérir les couples d'entrées/sorties-désirées à partir de mesures expérimentales, ou bien par la simulation d'un modèle de référence aussi précis que possible, s'il n'est pas envisageable de faire des mesures.

Pour faire l'acquisition des couples d'entrées/sorties qui serviront à l'apprentissage du réseau, nous faisons varier les paramètres d'entrée sur toute leur gamme de fonctionnement et nous obtenons les réponses du système réel ou de notre modèle de référence. En théorie, la variation des stimuli doit prendre en compte toutes les valeurs et les évaluations possibles au niveau des entrées. Cependant, si il y a un grand nombre de stimuli ou bien si un stimulus est défini par plusieurs paramètres (par exemple, un signal carré défini par son amplitude, sa fréquence, son temps de montée et de descente), il est difficile de prévoir tous les cas possibles. Par ailleurs, un apprentissage exhaustif prenant en compte toutes

les variations des stimuli peut être lourd et non-efficace. Il est donc souvent préférable de figer les paramètres les moins importants selon le type de système et l'application à modéliser. Par exemple, pour modéliser la diaphonie entre deux lignes de bus adjacentes, si les couples d'entrées/sorties-désirées sont les tensions d'entrées/sorties-désirées des deux lignes, nous pouvons faire varier les deux tensions d'entrées de façon exhaustive en jouant sur la fréquence, l'amplitude, etc. Cependant, dans le cas de la transmission numérique, les valeurs des tensions correspondant aux niveaux logiques "0" et "1" sont connues et fixes, et les temps de montée/descente des signaux sont généralement spécifiés par la norme employée. Il est donc inutile de faire varier ces paramètres de façon exhaustive. Ainsi, on allège l'apprentissage du réseau et on se concentre plutôt sur les fréquences de variation des signaux.

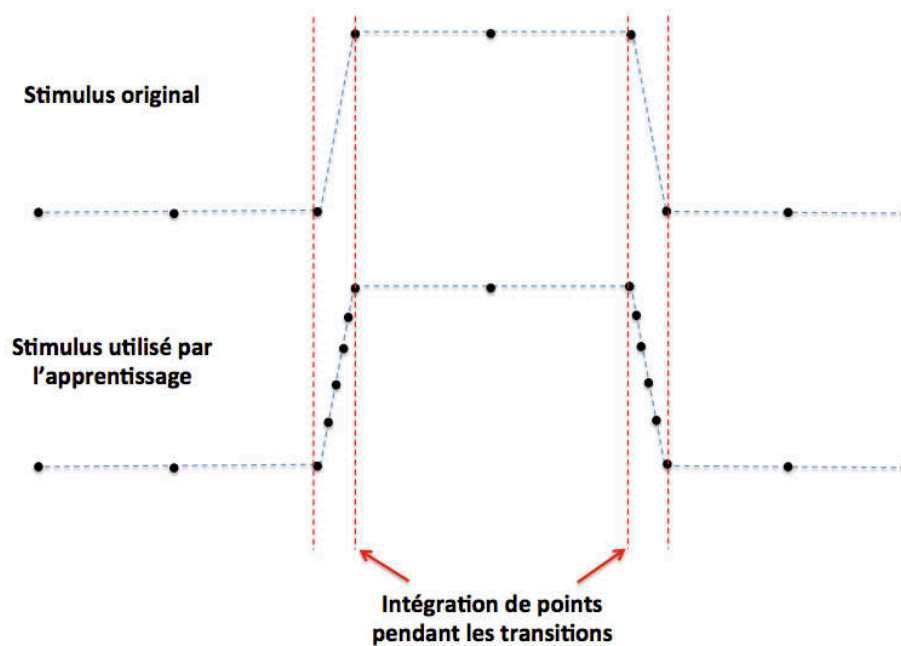


FIGURE 3.22 – Principe d'insertion de valeurs.

Les stimuli et les résultats sont stockés dans un fichier qui servira dans la deuxième étape. Pour faciliter l'apprentissage du réseau de neurones dans l'étape suivante, nous pouvons insérer quelques valeurs entre les niveaux logiques "0" et "1" pendant l'enregistrement des stimuli, comme l'illustre dans la figure 3.22. Un exemple de fichier de stockage est présenté dans la figure 3.23.

3.4.3 Etape 2 - Apprentissage du réseau de neurones

La deuxième étape est l'apprentissage du réseau de neurones pour modéliser les phénomènes d'intégrité du signal. A l'aide du réseau, nous approximations par une fonction mathématique les réponses du système en fonction des entrées et conformément aux sorties obtenues au cours de la première étape. Cette fonction représente les phénomènes d'intégrité du signal propres à notre système.

Dans notre thèse, cette deuxième étape est réalisée par le *Neural Network Toolbox* de MATLAB [DBH09]. A l'aide de MATLAB, la procédure d'apprentissage du réseau de

Stimuli			Résultats				
Entrée 1	Entrée 2	...	Entrée n	Sortie 1	Sortie 2	...	Sortie n
0	1		0	0.8132	0.814		0.810
0	0.91		0.0005	0.8132	0.814		0.810
0.18	0.73		0.001	0.8128	0.8139		0.811
0.36	0.4		0.002	0.8128	0.8139		0.811
...	...		...	...	...		...
1	0		0.95	0.8127	0.8137		0.813
0.9	0		1	0.8127	0.8137		0.813

FIGURE 3.23 – Fichier de stockage des couples d'entrées/sorties-désirées.

neurones est réalisée automatiquement. Cependant, l'architecture du réseau de neurones doit être spécifiée manuellement par les concepteurs avant la procédure d'apprentissage. Comme on l'a vu précédemment, l'architecture du réseau dépend des phénomènes que notre bloc non-fonctionnel doit être capable de représenter.

Comme nous l'avons écrit dans la section 3.2.2.4, nous allons choisir une architecture de réseau de neurones à trois couches. Le choix d'un réseau non-bouclé ou bouclé dépend des éléments à modéliser. Le *Neural Network Toolbox* de MATLAB propose plusieurs fonctions qui implémentent les différentes structures d'architectures. Par exemple (cf. figure 3.24), FTDNN (*Focused Time-Delay Neural Network*) et TDNN (*Distributed Time-Delay Neural Network*) sont des réseaux multicouches non-bouclés avec délais, NARX network (*Nonlinear Autoregressive exogenous network*) et Layer-Recurrent network sont des réseaux bouclés. Nous pouvons choisir une architecture parmi celles-ci selon les différents cas d'étude. Pour les fonctions d'activation, nous choisissons le tan-sigmoïde (équation 3.5) pour la couche cachée et le purelig (équation 3.6) pour la couche de sortie.

Ensuite, il faut définir les autres paramètres de configuration de l'architecture. Nous commençons par le nombre d'entrées et le nombre de sorties. Evidemment, ces paramètres correspondent aux stimuli et aux résultats obtenus dans la première étape. Pour un réseau de neurones à trois couches, il faut encore fixer le nombre de neurones dans la couche cachée. Pour fixer cette valeur, on prend un nombre initial de neurones en se basant sur une des heuristiques du tableau 3.3, puis on applique les critères suivants :

1. Si l'erreur du réseau au cours de la phase d'apprentissage est faible (selon les attentes du concepteur pour ce cas d'étude), mais que l'erreur des sorties lors de la phase de validation est grande, la flexibilité du réseau de neurones est mauvaise. Il faut alors réduire le nombre de neurones dans la couche cachée.
2. Si l'erreur du réseau au cours de la phase d'apprentissage est grande, il faut augmenter le nombre de neurones dans la couche cachée.
3. Si tous les poids de connexions entre les neurones sont du même ordre de grandeur, alors il faut augmenter le nombre de neurones dans la couche cachée.

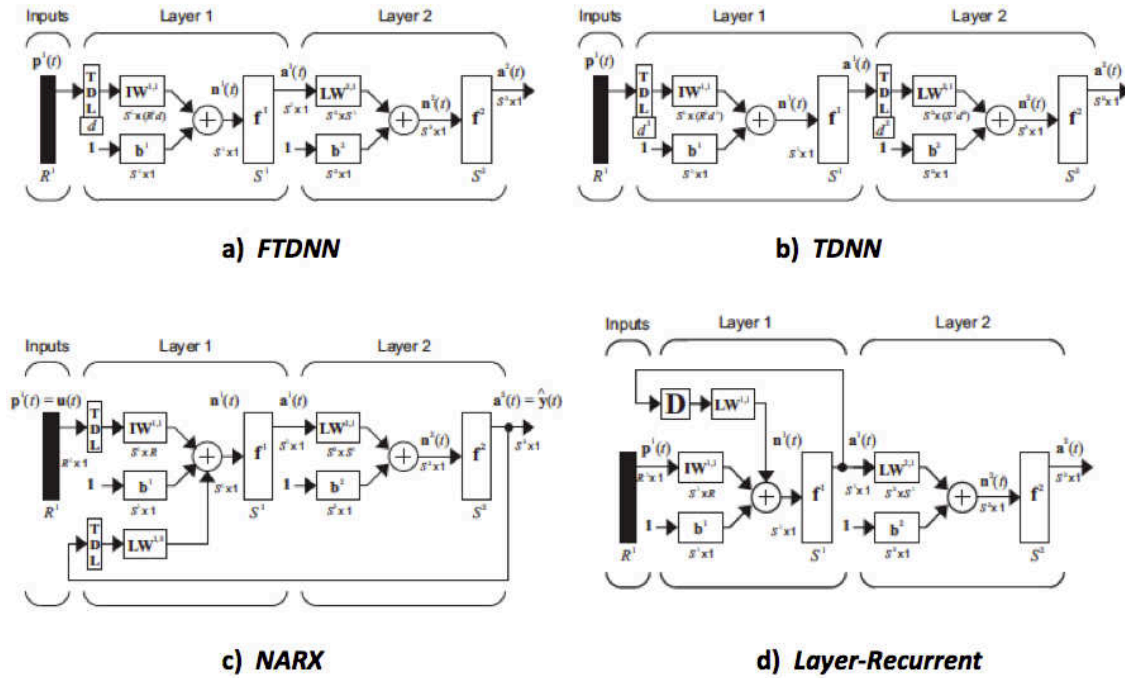


FIGURE 3.24 – Architectures de réseaux de neurones proposés par le *Neural Network Toolbox* de MATLAB.

Dans notre thèse, le nombre de neurones dans la couche cachée est déterminé de façon empirique en se basant sur ces critères. Le nombre initial est déterminé par l'équation $N > 3N_i$ [Hus89], où N est le nombre de neurones dans la couche cachée, et N_i est le nombre d'entrées.

La *toolbox* de MATLAB définit par ailleurs un paramètre *epoch* qui fixe le nombre de mises à jour des poids et des biais qui seront effectuées au cours de la phase d'apprentissage. De plus, pour certains cas d'étude, il faut insérer des unités de retard pour les entrées ou les éventuelles boucles de réaction du réseau. Ce nombre de délais et la valeur de *epoch* sont déterminés empiriquement.

Une fois que la structure d'architecture du réseau de neurones, les fonctions d'activation et les paramètres de configuration sont définis, nous pouvons lancer l'apprentissage dans MATLAB.

A la fin de la deuxième étape, nous extrayons les valeurs des poids et les valeurs de biais du réseau de neurones et nous enregistrons ces valeurs et les autres paramètres du réseau dans un fichier d'extension ".nn" qui sera utilisé dans la troisième étape, pour reconstruire le réseau de neurones en SystemC-AMS. Un exemple de ce fichier est présenté dans la figure 3.25. Ce fichier décrit un réseau bouclé à une entrée, une sortie, quatre neurones dans la cache couchée et deux délais pour la boucle de rétroaction.

3.4.4 Etape 3 - Construction du modèle SystemC-AMS TDF implémentant le réseau de neurones et les adaptateurs

La troisième étape a deux objectifs :

```

*****
***Nombre d'entrées (Ni)***
1
*****
***Nombre de sorties (No)***
1
*****
***Nombre de neurones dans la couche cachée (N)***
4
*****
***Nombre de data à traiter à chaque pas (Nt)***
***Non-bouclé : Nt=Ni+Ni×Nid***
***Bouclé : Nt=Ni+Ni×Nid+No×Nrd***
***Nid : Nombre de délai d'entrée (0)***
***Nrd : Nombre de délai de rétroaction (2)***
3
*****
*** Matrice IW : matrice de poids entre l'entrée***
*** et la couche cachée***
-1.4824      1.25700      0.01070
1.26930      -0.7326      0.61650
-0.2965      -0.6033      0.01070
-1.7441      0.05530      -1.1170
*****
***Matrice LW : matrice de poids entre la couche cachée***
*** et la couche sortie***
-0.5831      0.22660      0.18550      0.01710
*****
***Matrice b***
2.31650
-1.2829
0.90170
0.10010
*****
***Valeurs d'initialisation des données à traiter***
0      0      0
*****
***Min/Max des couples d'entrées/sorties***
-3.8579
5.49210
-3.8579
5.49210

```

FIGURE 3.25 – Forme générale d'un fichier ".nn".

1. Construire le module SystemC-AMS TDF qui permet de représenter les performances de l'intégrité du signal à l'aide du réseau de neurones.

2. Développer les adaptateurs pour associer le bloc non-fonctionnel au bloc fonctionnel.

3.4.4.1 Implémentation du réseau de neurones

Pour le premier objectif, nous avons créé une classe en C++ "neuralnetwork" pour reconstruire le réseau de neurones à partir du fichier ".nn". Lorsque l'on déclare un nouvel objet de la classe "neuralnetwork", le constructeur fait appel à un fichier ".nn" spécifié par l'utilisateur, crée un réseau de neurones à partir de ce fichier et établit la fonction d'approximation de ce réseau. Cela est effectué dans la méthode *initialize()* du module TDF. Une fois que le réseau de neurones est reconstruit, le classe dispose d'une méthode *sim()*. Cette méthode que nous avons développée permet d'émuler le fonctionnement du réseau de neurones : acquisition des signaux d'entrées, calcul des signaux de sorties à l'aide de la fonction d'approximation. La méthode *sim()* de la classe "neuralnetwork" est appelée dans la description TDF du bloc non-fonctionnel. Elle va ainsi déterminer les performances d'intégrité du signal du composant. La figure 3.26 montre un exemple général du code SystemC-AMS pour la construction de réseau de neurones. A chaque cycle de simulation, la valeur de *input* est mise à jour et la valeur de *output* est calculée à l'aide de la méthode *sim()*.

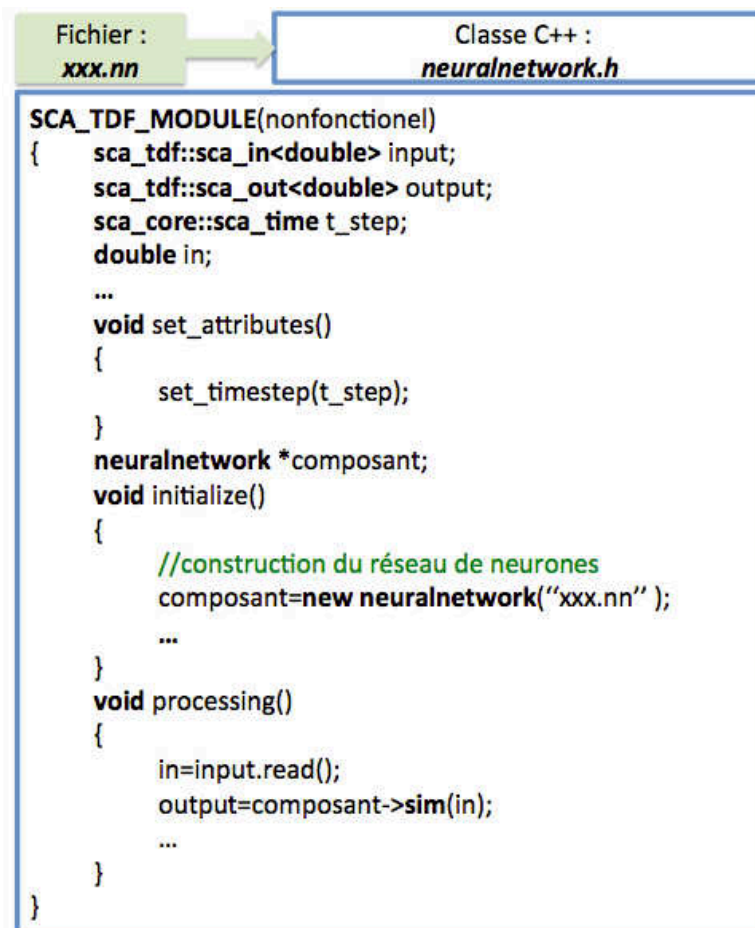


FIGURE 3.26 – Exemple général d'un bloc non-fonctionnel.

3.4.4.2 Adaptateurs

Dans notre bloc non-fonctionnel, nous analysons les performances d'intégrité du signal sous forme analogique. Mais, les entrées/sorties de notre bloc fonctionnel avec lequel le bloc non-fonctionnel s'interface sont présentées sous forme numérique. Pour adapter les signaux entre les deux blocs, il faut à présent développer des adaptateurs.

Du bloc non-fonctionnel au bloc fonctionnel, l'adaptateur va transformer des signaux de forme analogique en signaux de forme binaire. Nous réalisons cette opération à l'aide d'un détecteur de seuil. Deux tensions de seuil, V_h et V_l , sont définies. A chaque activation du module, la sortie du réseau de neurones est comparée à ces seuils. Si la sortie dépasse V_h , on génère un niveau logique haut ; si la sortie n'atteint pas V_l , on génère un niveau logique bas. Ce détecteur de seuil est présenté dans la figure 3.27.

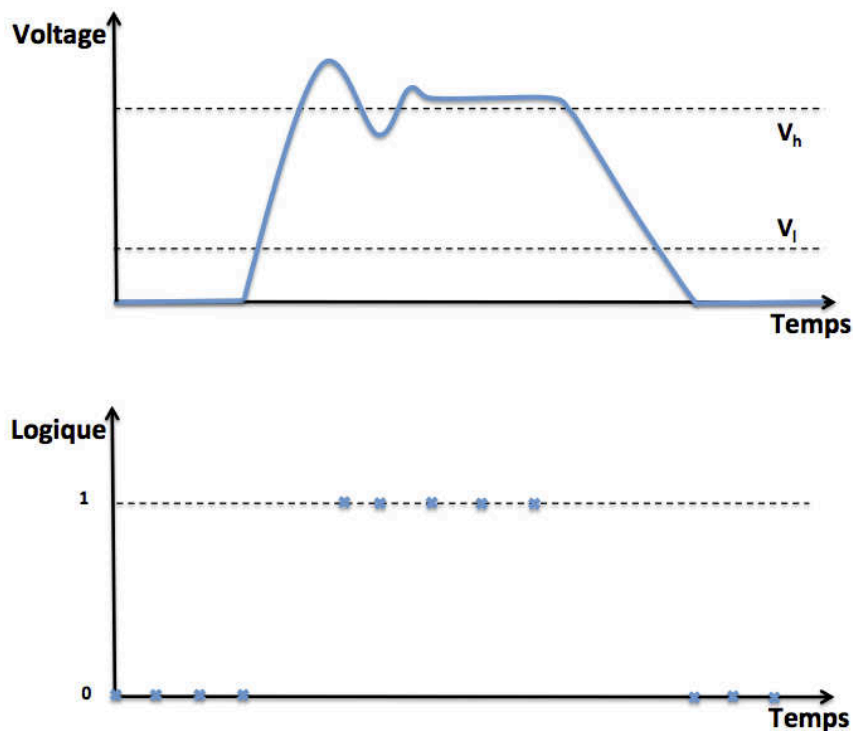


FIGURE 3.27 – Principe de détection de seuil par les adaptateurs.

Du bloc fonctionnel au bloc non-fonctionnel, la transformation du signal numérique en signal analogique peut être réalisée directement par notre réseau de neurones. Cependant, pour faciliter l'apprentissage, nous avons inséré dans l'étape 1 des valeurs entre les niveaux logiques "0" et "1" (comme le montre la figure 3.22). Nous ajoutons donc, entre le bloc fonctionnel et le bloc non-fonctionnel, un adaptateur dont le rôle est d'insérer des valeurs de façon similaire à ce qui a été réalisé dans l'étape 1, pour renforcer la cohérence entre l'apprentissage et la simulation. Pour ne pas ajouter artificiellement un délai au signal, cet adaptateur n'ajoute pas de nouveaux points, mais plutôt, lorsqu'il a détecté une transition, remplace les valeurs binaires de quelques points par des valeurs intermédiaires, comme le montre la figure 3.28.

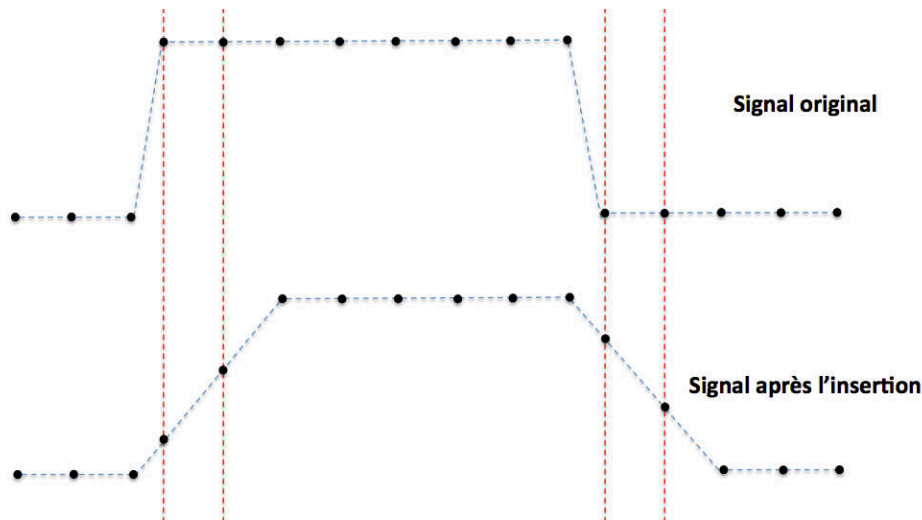


FIGURE 3.28 – Principe d'insertion de valeurs intermédiaires par les adaptateurs.

3.4.5 Conclusion

Le bloc non-fonctionnel permet de représenter toutes les performances de l'intégrité du signal du système à un haut niveau d'abstraction.

Pour réaliser ce bloc, nous avons procédé en trois étapes. Les deux premières étapes servent à créer un réseau de neurones qui permet de modéliser les non-linéarités et l'évolution dynamique des signaux, liées aux problèmes d'intégrité du signal. Dans la dernière étape, ce réseau de neurones est reconstruit dans un module SystemC-AMS TDF. Des adaptateurs, permettant de connecter ce module TDF avec le bloc fonctionnel, sont également adjoints dans cette étape. Le tout est implémenté en SystemC-AMS TDF. Ainsi, il est possible de les associer et de les simuler avec des blocs fonctionnels, développés en SystemC/SystemC-AMS.

Dans la section suivante, nous discuterons des avantages potentiels de cette méthode de modélisation de l'intégrité du signal.

3.5 Conclusion

Dans ce chapitre, nous avons proposé une méthodologie de modélisation de l'intégrité du signal à haut niveau d'abstraction basée sur l'association de blocs non-fonctionnels et de blocs fonctionnels.

Le bloc non-fonctionnel est développé sur la base d'un réseau de neurones en SystemC-AMS TDF. L'emploi d'un réseau de neurones se justifie par la nécessité de modéliser des phénomènes non linéaires et dynamiques de l'intégrité du signal à un haut niveau d'abstraction. Des adaptateurs en SystemC-AMS TDF ont aussi été implémentés afin de connecter ces blocs non-fonctionnels à des blocs fonctionnels. Ces blocs fonctionnels, eux aussi développés en SystemC/SystemC-AMS, modélisent les fonctionnalités idéales du système indépendamment du bloc non-fonctionnel. En les associant avec les blocs non-fonctionnels, un modèle complexe est donc établi et l'on peut modéliser le comportement du système dans son ensemble et analyser ses performances d'intégrité du signal à un haut

niveau d'abstraction.

Cette méthodologie présente potentiellement trois principaux avantages :

- A l'aide du réseau de neurones, le bloc non-fonctionnel permet de modéliser les performances d'intégrité du signal, avec toutes les évolutions non linéaires et dynamiques qui en découlent.
- A l'aide de l'apprentissage supervisé, la construction du bloc non-fonctionnel a besoin de moins d'informations sur le système. Elle n'a notamment pas besoin des informations confidentielles sur les technologies de conception. Cela simplifie sa réalisation et protège les informations sensibles du système.
- A l'aide de SystemC/SystemC-AMS et de réseau de neurones, les fonctions du système et les performances de l'intégrité du signal sont modélisées et simulées ensemble à haut niveau d'abstraction. Alors, notre méthodologie doit permettre de diminuer le temps de simulation.

Dans le chapitre suivant, nous appliquons notre méthodologie à deux cas d'études : un système communiquant via un bus I²C et un système communiquant via un bus USB 3.0.

Chapitre 4

Simulation et Résultats

Sommaire

4.1	Introduction	64
4.2	Première Application : système communiquant via un bus I²C	64
4.2.1	Protocole I ² C	64
4.2.1.1	Généralités sur I ² C	65
4.2.1.2	Protocole de données	65
4.2.2	Présentation du système à modéliser et du modèle de référence .	67
4.2.2.1	Système à modéliser	67
4.2.2.2	Modèle de référence	68
4.2.3	Développement de notre modèle	70
4.2.3.1	Blocs Fonctionnels	70
4.2.3.2	Bloc non-fonctionnel	71
4.2.4	Résultats de simulation	75
4.2.4.1	Introduction	75
4.2.4.2	Résultats et analyse	75
4.2.5	Conclusion	79
4.3	Deuxième Application : système communiquant via un bus USB 3.0	80
4.3.1	Protocole USB 3.0	81
4.3.1.1	Généralités	81
4.3.1.2	Architecture de la liaison SuperSpeed	82
4.3.1.3	Protocole de la liaison SuperSpeed	83
4.3.2	Présentation du système à modéliser et du modèle de référence .	84
4.3.2.1	Système à modéliser	84
4.3.2.2	Modèle de référence	86
4.3.3	Développement de notre modèle	88
4.3.3.1	Introduction	88
4.3.3.2	Bloc non-fonctionnel	88
4.3.4	Résultats de simulation	94
4.3.4.1	Introduction	94
4.3.4.2	Résultats et analyse	94
4.3.5	Conclusion	103
4.4	Conclusion	103

4.1 Introduction

Dans le chapitre précédent, pour répondre à notre problématique, nous avons proposé une méthodologie de modélisation de l'intégrité du signal au niveau système. Pour valider cette méthodologie, nous allons à présent l'appliquer à deux cas d'études :

1. Des systèmes communiquant via un bus I²C : une application autour d'un bus I²C va nous permettre de démontrer la faisabilité de notre méthodologie et d'évaluer les apports des moteurs de simulation TDF conventionnel et dynamique.
2. Des systèmes communiquant via une liaison USB 3.0 : ce cas d'étude va nous permettre d'étendre notre méthodologie à des applications à haute fréquence, et à montrer sa flexibilité (paramétrage des modèles, modularité en appliquant un réseau de neurones par sous-bloc du système).

Dans ce chapitre, nous présentons ces deux cas d'études en détaillant le développement des modèles, la simulation et l'analyse des résultats. La deuxième section est centrée sur l'application autour du bus I²C. Dans la troisième section, nous présentons le cas d'étude autour de l'USB 3.0. Dans les deux sections, les applications sont décrites de façon similaire : nous présentons d'abord le protocole de communication utilisé dans l'application. Ensuite, nous présentons l'application à modéliser. Nous décrivons également un modèle de référence de l'application qui nous sert à valider notre méthodologie. Après cela, le développement de notre modèle est présenté. Enfin, nous montrons les résultats de simulation de nos modèles et nous les comparons avec le modèle de référence. A partir de cette comparaison, nous analysons les performances (en termes de précision et de temps de simulation) de notre méthodologie.

4.2 Première Application : système communiquant via un bus I²C

Nous présentons dans cette section l'application de notre méthodologie à un système communiquant via un bus I²C [npx]. Cette application nous permet de démontrer la faisabilité de notre méthodologie. De plus, deux types de bloc non-fonctionnel, l'un basé sur le TDF conventionnel et l'autre basé sur le TDF dynamique, sont développés et comparés pour évaluer les avantages du TDF dynamique.

Cette section comporte cinq parties. Nous présentons le protocole I²C dans la section 4.2.1. Ensuite, le système à modéliser est présenté dans la section 4.2.2, ainsi qu'un modèle de référence de cette application. Le développement de nos blocs fonctionnels et non-fonctionnels est décrit en section 4.2.3. La section 4.2.4 présente et analyse les résultats de simulation de notre modèle. Nous terminons cette partie par une conclusion.

4.2.1 Protocole I²C

Le bus I²C (Inter Integrated Circuit) [npx] est un bus série, développé par Philips au début des années 80 pour minimiser l'encombrement des liaisons entre les circuits intégrés pour des applications de domotique ou d'électronique domestique. Parmi les bus de terrain existants, I²C propose un protocole simple à implanter. Il permet ainsi d'accéder

facilement aux différents circuits d'une application. Dans un ordinateur personnel, par exemple, le protocole I²C va être utilisé par le SMBus (*System Management Bus*) de la carte mère et le DDC (*Display Data Channel*) du moniteur. Il existe d'innombrables périphériques communiquant via ce protocole simple et rapide. D'ailleurs, de très nombreux microcontrôleurs proposent une interface de gestion d'un bus I²C.

4.2.1.1 Généralités sur I²C

Le bus I²C permet de faire communiquer entre eux des composants électroniques grâce à seulement deux fils bidirectionnels :

- SDA (*Serial Data Line*) : ligne de données.
- SCL (*Serial Clock Line*) : ligne d'horloge de synchronisation.

On ajoute à ces deux lignes une ligne de masse qui doit être commune aux équipements. Ce mode d'échange de données en série facilite la réalisation des circuits imprimés au niveau des interconnexions. En revanche, la sérialisation du bus limite la vitesse de transfert des données. Dans [nxp], le bus I²C propose cinq modes de transmission différents avec les débits suivants : 100 kbit/s en *Standard-mode*, 400 kbit/s en *Fast-mode*, 1 Mbit/s en *Fast-mode Plus*, 3.4 Mbit/s en *High-speed mode* et 5 Mbit/s en *Ultra Fast-mode*. I²C peut être utilisé dans les applications pour lesquelles la vitesse de transmission n'est pas un élément crucial.

Les composants connectés au bus I²C peuvent être de deux types :

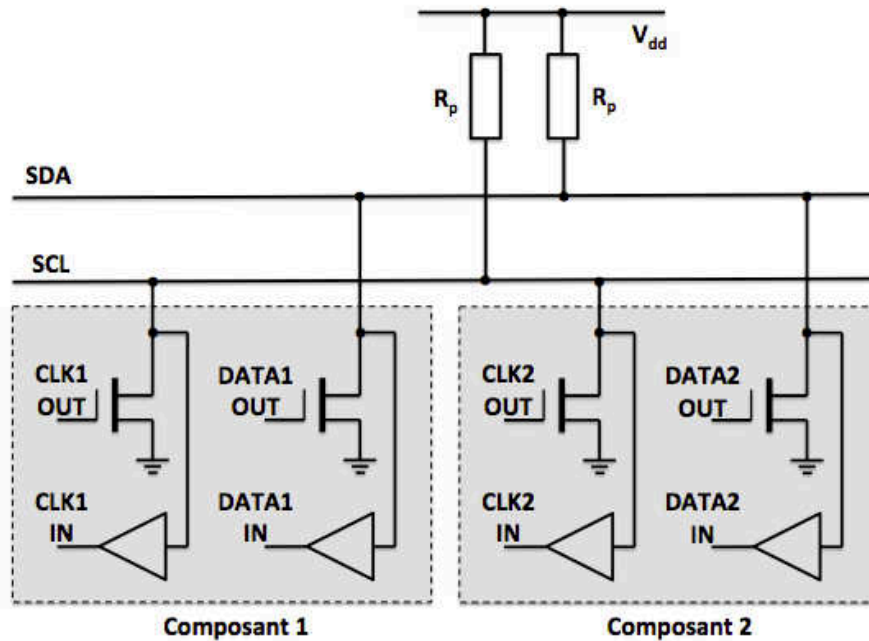
- Maître : nœud qui génère l'horloge et initie la communication avec des esclaves.
- Esclave : nœud qui reçoit l'horloge et répond uniquement s'il est sollicité par un maître.

Le nombre de composants qu'il est possible de relier est limité par l'espace d'adressage et également par la charge capacitive des lignes SDA et SCL (400 pF au total). Le bus I²C a la capacité de fonctionner avec plusieurs maîtres (mode multi-maîtres).

Les entrées/sorties d'un circuit I²C sont implantées en mode collecteur ouvert dans les circuits TTL, ou drain ouvert dans les circuits CMOS. Le niveau du bus est alors défini par la fonction *wired-and* de toutes les entrées connectées. La structure des entrées/sorties est présentée dans la figure 4.1. Il y a deux états différents : le lecture et l'écriture. En lecture, l'interface d'E/S fonctionne comme un détecteur de seuil. Elle traduit la tension sur la ligne SDA ou SCL en un niveau logique par comparaison avec une tension de seuil. Dans l'état d'écriture, les interfaces fonctionnent comme un collecteur ou un drain ouvert pour déterminer l'arbitrage entre les différents nœuds, et adapter les signaux qui sortent du bloc numérique aux tensions en vigueur sur le bus. Si aucune communication n'intervient sur le bus, le niveau des deux lignes SDA et SCL est fixé à la tension V_{dd} , grâce à des résistances de rappel (représentées par R_p dans la figure 4.1).

4.2.1.2 Protocole de données

Le protocole I²C définit la succession des états possibles sur le bus (SDA et SCL) et comment les circuits connectés au bus réagissent en cas de conflits. La figure 4.2 présente l'allure d'une trame de données au format I²C. La communication commence par une condition de départ (START) et se termine par une condition d'arrêt (STOP). Les condi-

FIGURE 4.1 – Schéma électrique des E/S du bus I²C (R_p : Résistance de rappel).

tions de départ et d'arrêt sont uniquement générées par le maître (figure 4.3). Avant de tenter de prendre le contrôle du bus, il faut que celui-ci soit libre. La condition de départ est caractérisée par le passage de la ligne SDA du niveau haut au niveau bas, pendant que la ligne SCL est maintenue au niveau haut. A partir de ce moment, le bus est occupé et aucun autre maître ne peut prendre le contrôle du bus. La condition d'arrêt signifie la fin d'une trame et libère le bus. Elle est réalisée en changeant l'état de la ligne SDA du niveau bas au niveau haut pendant que la ligne SCL est maintenue au niveau haut.

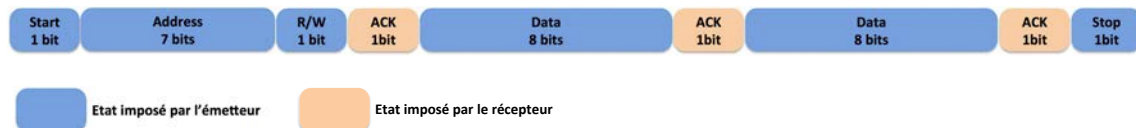
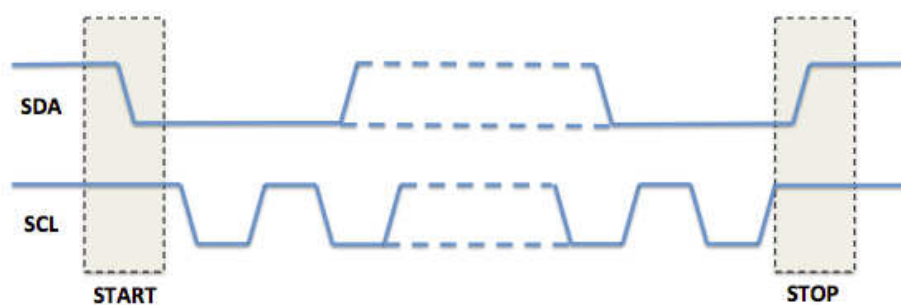


FIGURE 4.2 – Trame de donnée I²C.

Exceptées les conditions de départ et d'arrêt, toutes les autres données émises sur les deux lignes SDA et SCL sont par paquets d'octets. L'octet émis commence par le bit de poids fort au rythme d'un bit à chaque période d'horloge de SCL. Les données sur la ligne SDA doivent être stables pendant le niveau haut de SCL, sinon cela pourrait être interprété comme une condition de départ ou d'arrêt.

Le bit START est suivi par l'envoi d'une adresse qui caractérise chaque composant connecté au bus I²C. Cette adresse a une taille de 7 bits ou 10 bits en fonction du mode de transmission choisi. Nous utiliserons des adresses sur 7 bits. L'octet d'adressage est dans ce cas divisé en deux champs :

- (i) Les 7 premiers bits correspondent à l'adresse de la cible (destination du message).
- (ii) Le dernier bit est le bit R/W :

FIGURE 4.3 – Conditions START et STOP du protocole I²C.

- S'il est à 0, le maître indique qu'il souhaite écrire des données, et donc que l'esclave doit les lire.
- S'il est à 1, le maître indique qu'il veut lire des données, et donc que l'esclave doit les lui envoyer.

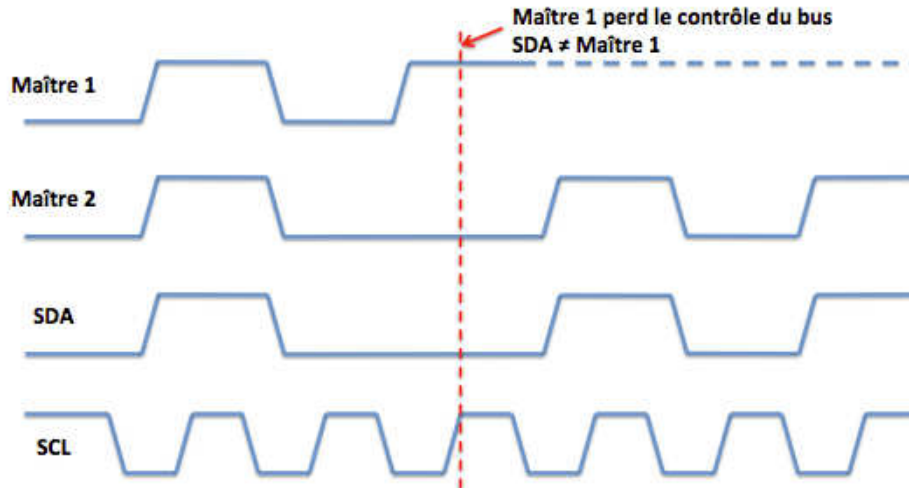
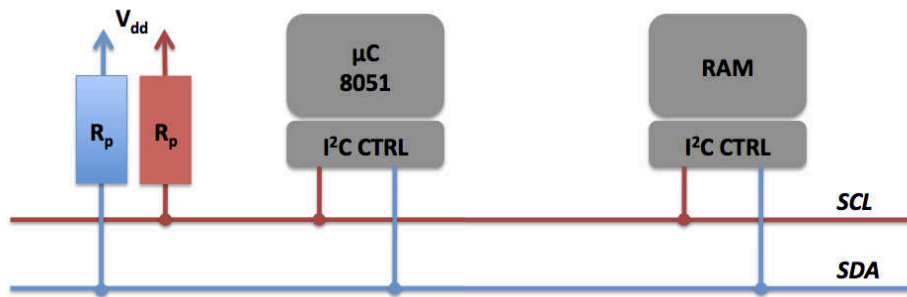
L'envoi d'un octet se termine obligatoirement par un bit d'acquittement (ACK). A la fin de l'octet, l'émetteur libère la ligne SDA. Si le récepteur a bien reçu les données, il doit alors baisser la ligne SDA et la maintenir au niveau bas pendant le niveau haut de la ligne SCL. Dans le cas où l'émetteur ne reçoit pas ce bit d'acquittement, le transfert est alors abandonné et doit être éventuellement réitéré.

Comme nous l'avons indiqué dans la section précédente, le bus I²C peut fonctionner avec plusieurs maîtres. Dans ce cas, la priorité est donnée au nœud maître qui initie la communication en premier. En cas de demandes de transmission simultanées, l'arbitrage est effectué en fonction des messages transmis. Tant que les maîtres émettent le même bit, ils peuvent continuer à transmettre. Si les bits émis sont différents, le nœud qui émet un "0" est prioritaire sur le nœud qui émet un "1". Ce dernier cesse alors d'émettre son message. On dit donc que l'état logique "0" est l'état dominant et que l'état logique "1" est l'état récessif. Cette procédure d'arbitrage illustrée dans la figure 4.4 permet ainsi d'éviter les conflits : le maître 1 perd le contrôle du bus parce qu'il impose un "1" pendant que le maître 2 impose un "0". La procédure d'arbitrage est automatiquement réalisée par l'implémentation physique des nœuds (collecteur ou drain ouvert).

4.2.2 Présentation du système à modéliser et du modèle de référence

4.2.2.1 Système à modéliser

L'architecture de la plate-forme autour d'un bus I²C est donnée dans la figure 4.5. Elle comprend un nœud maître, un microcontrôleur 8051 [Phi99], et un nœud esclave, une mémoire RAM. Le code embarqué dans le microcontrôleur définit le comportement de cette plate-forme. Dans notre cas, un code simple est exécuté pour effectuer des lectures ou des écritures de la mémoire RAM via le bus I²C. Chaque nœud du système dispose d'un contrôleur de bus I²C pour interfacer le nœud avec le bus. Le contrôleur traduit les requêtes ou données des nœuds et les met en forme selon le protocole I²C.

FIGURE 4.4 – Procédure d’arbitrage sur le bus I²C.FIGURE 4.5 – Exemple de système communiquant via un bus I²C.

4.2.2.2 Modèle de référence

Pour notre application, nous utilisons, comme modèle de référence, les travaux de M. Alassir [Ala10].

Il modélise le bus I²C en deux parties : une partie numérique et une partie analogique. La partie numérique représente le contrôleur de bus. Elle est décrite en SystemC, au niveau CABA (pour *Cycle Accurate Bit Accurate*). La partie analogique représente les lignes de bus et les E/S des composants. Elles sont modélisées en SystemC-AMS, par un module ELN. Dans ce module ELN, le collecteur ou drain ouvert réalisant l’écriture sur le bus est représenté par un simple réseau électrique comme le montre la figure 4.6. Le modèle de transistor n’étant pas disponible en SystemC-AMS ELN, il a donc été remplacé par un interrupteur, une résistance et une capacité. Les lignes SDA et SCL sont modélisées par un réseau RLC organisé en Π (cf. figure 2.8.b). Cette méthode de modélisation est à bas niveau d’abstraction en raison de l’utilisation de modèles électriques (ELN) pour représenter l’intégrité du signal.

Cette méthode a été validée expérimentalement par mesure d’un système réel développé par M. Alassir. Cette comparaison montre que son modèle a une bonne précision (l’erreur

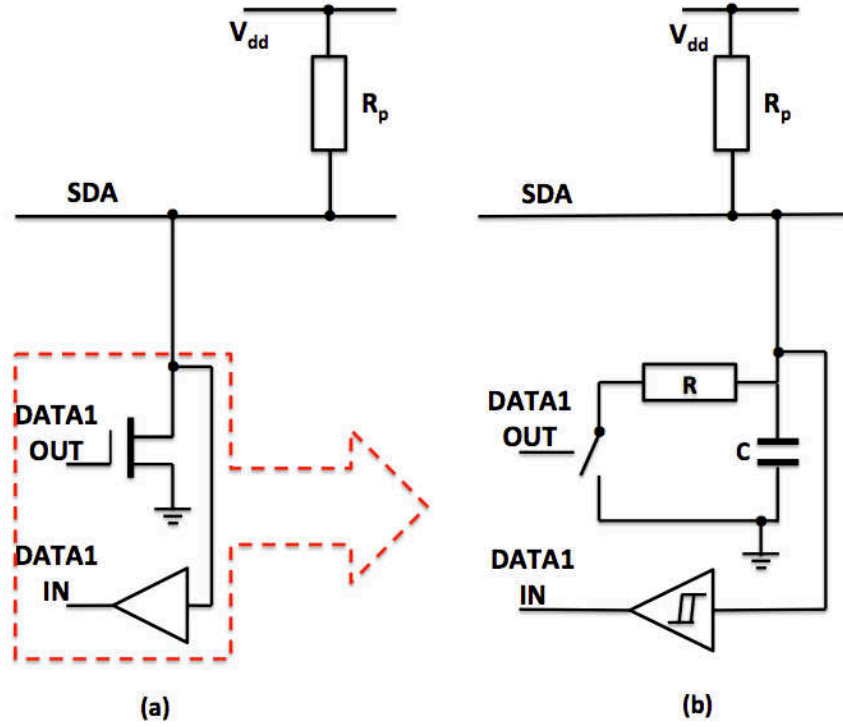


FIGURE 4.6 – (a) Schéma original de SDA (b) Schéma implémenté en SystemC-AMS ELN [ADR⁺08].

moyenne est d'environ 3%) [ADRG13]. La figure 4.7 présente les résultats de simulation du modèle de référence. Dans cette figure, il y a deux types de phénomènes représentés sur les lignes SDA et SCL :

- Temps de réponse : Sur les fronts montants et descendants des signaux, nous observons un temps de réponse. Ce phénomène correspond à la constante de temps d'un circuit RC. Dans notre cas, il est causé par les résistances et les capacités équivalentes existantes dans le réseau ELN de notre modèle (cf. figure 4.6.b et figure 2.8.b). Ce temps de réponse est défini par la formule suivante (équation 4.1) :

$$\tau = R \times C. \quad (4.1)$$

Dans notre système, ces deux valeurs dépendent du type de matériel, de la longueur des lignes de transmission, de la tension d'alimentation ou du nombre de composants connectés [Phi00]. Dans [nxp], les temps de montée et de descente sont limités pour les cinq modes (cf. section 4.2.1.1).

- La diaphonie entre les lignes SDA et SCL : Un changement d'état sur l'une des lignes du bus va interférer avec l'état de l'autre ligne. Ce phénomène est la diaphonie entre les lignes de transmission adjacentes SDA et SCL. Ce phénomène est causé principalement par l'inductance mutuelle ou la capacitance mutuelle entre les deux lignes [Bog03]. La transmission I²C s'effectue à des fréquences relativement faibles, la contribution de l'inductance mutuelle peut alors être négligée. L'intensité du phénomène dépend de la soudaineté des transitions du signal perturbant, de la distance entre les deux lignes de transmission et de la longueur de ces lignes.

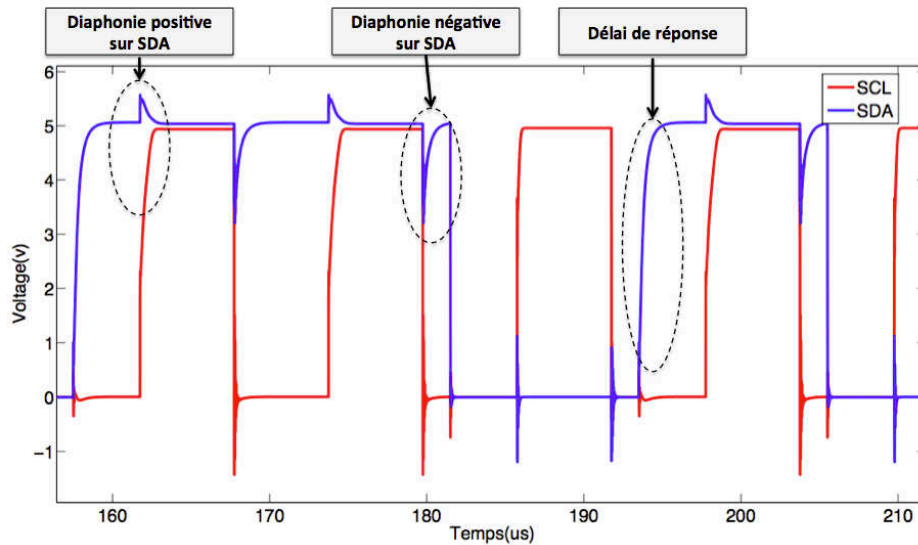


FIGURE 4.7 – Résultats de simulation du modèle de référence.

4.2.3 Développement de notre modèle

Notre modèle a pour objectif de représenter les fonctionnalités, ainsi que l'intégrité du signal sur la plate-forme de la figure 4.5. Nous détaillons dans cette section la construction des blocs fonctionnels et non-fonctionnels de ce modèle. L'architecture principale de ce modèle est donnée dans la figure 4.8. A l'aide du réseau de neurones et du moteur de simulation TDF de SystemC-AMS, notre modèle va permettre d'accélérer la simulation, et en même temps, de garder une bonne précision.

4.2.3.1 Blocs Fonctionnels

Nos blocs fonctionnels sont utilisés pour représenter toutes les fonctions numériques du système. Pour l'application de la figure 4.5, ce bloc doit inclure les fonctions des deux nœuds (le microcontrôleur 8051 et la RAM), les contrôleurs de bus I²C ainsi que la fonction *wired-and* du bus.

Comme nous l'avons indiqué dans la section 4.2.2.1, un code simple est embarqué dans le microcontrôleur 8051. Il définit la fonctionnalité du système. Dans notre bloc fonctionnel, ce code est décrit en langage C.

Le contrôleur de bus I²C comporte trois sous-blocs (figure 4.9) :

- Interface microprocesseur : Ce sous bloc est basé sur une FIFO circulaire qui permet de transférer des données entre deux blocs n'ayant pas un débit instantané constant. Toutes les requêtes transmises par le microprocesseur sont enregistrées dans cette FIFO. Ce sous-bloc extrait les informations nécessaires (adresse, données, etc.) de la première requête, puis il initialise le sous bloc séquenceur. Dès qu'une requête a été traitée, la suivante est exécutée.
- Séquenceur : Il contient une machine d'état qui suit tous les états du protocole I²C. Toutes les opérations possibles sont implantées dans cette machine, notamment les

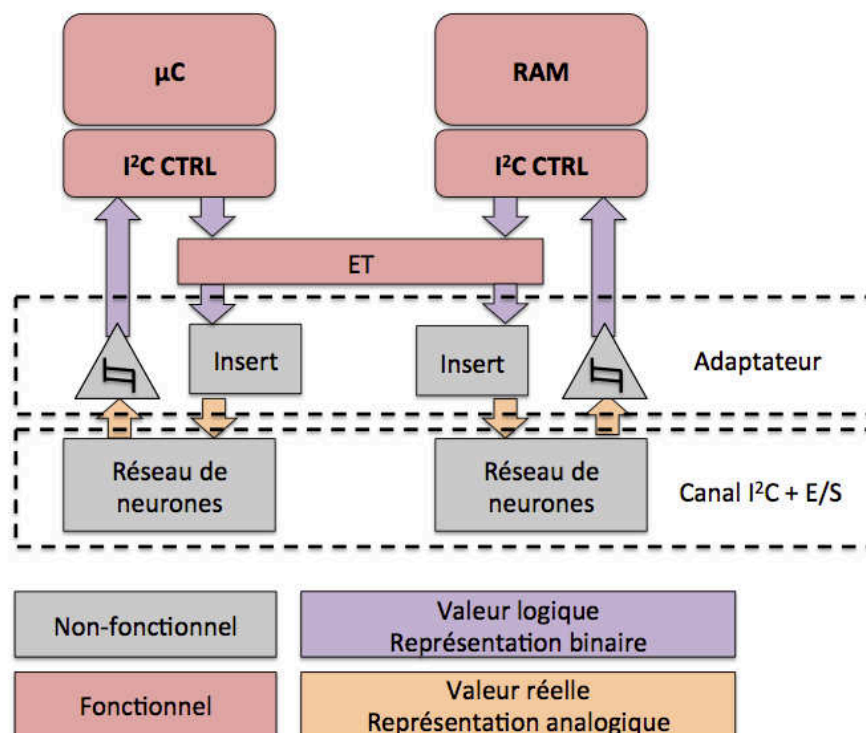


FIGURE 4.8 – Architecture du modèle de la plate-forme I²C (ET : Porte "ET"; Insert : Insertion).

opérations de lecture et d'écriture.

- Interface bus : Dans le mode émission, il sérialise l'octet reçu par le séquenceur pour qu'il soit envoyé sur le bus. Dans le mode réception, ce sous-bloc désérialise les données du bus et génère l'acquittement. Il génère également la fréquence de transmission des données à l'aide d'un diviseur d'horloge. Dans le mode multi-maîtres, il réalise l'arbitrage en comparant sa valeur théorique envoyée sur la ligne SDA avec la valeur réelle lue sur le bus. Pour simplifier l'écriture du modèle, les lignes bidirectionnelles SDA ou SCL sont chacune représentées par des signaux unidirectionnels (un signal en entrée pour la lecture, un signal en sortie pour l'écriture).

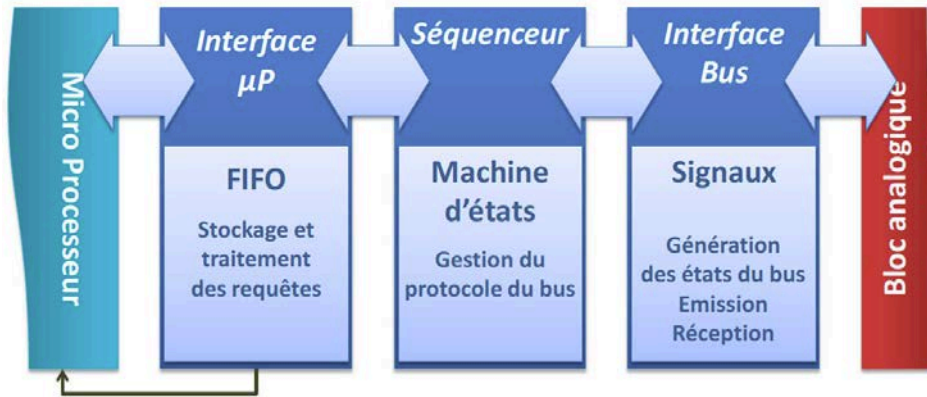
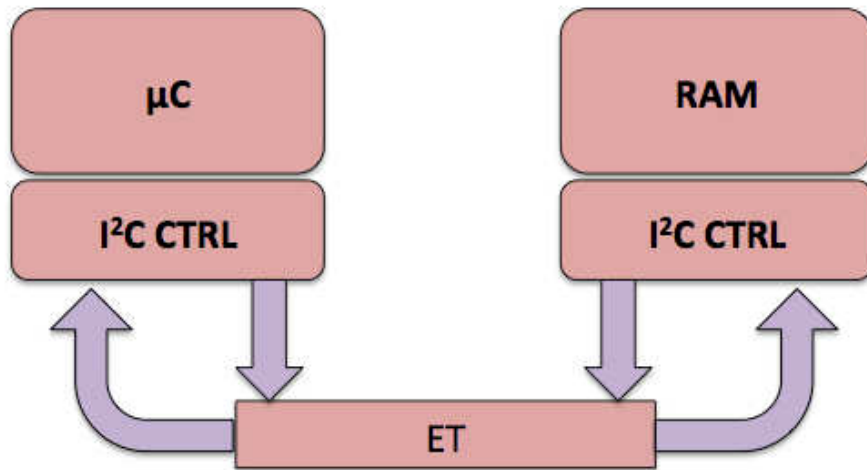
Dans notre thèse, le modèle du contrôleur est inspiré des travaux présentés dans [Ala10].

Enfin, une porte "ET" est ajoutée à notre bloc fonctionnel pour réaliser la fonction *wired-and* du bus I²C. A l'aide de cette porte "ET", le niveau logique du bus est fixé par le nœud qui impose un niveau dominant.

Nous pouvons connecter ensemble tous les blocs fonctionnels comme l'illustre la figure 4.10 et ainsi donner le comportement idéal de l'application. La simulation du bloc fonctionnel sera présentée et commentée dans la section 4.2.4.

4.2.3.2 Bloc non-fonctionnel

La partie non-fonctionnelle du modèle contient un module donnant les performances d'intégrité du signal de la plate-forme I²C, mais aussi les adaptateurs pour connecter les

FIGURE 4.9 – Bloc numérique du contrôleur de bus I²C [Ala10].FIGURE 4.10 – Bloc fonctionnel pour l'application I²C.

blocs fonctionnels avec ce module non-fonctionnel.

Dans le chapitre 3, nous avons indiqué qu'il faut 3 étapes pour bâtir notre bloc non-fonctionnel. Ici, dans la première étape, nous acquérons les couples d'entrées/sorties-désirées à partir d'un schéma électrique équivalent modélisé avec NGSPICE [NV13]. Ce circuit équivalent est présenté dans la figure 4.11. Il contient trois parties :

- (a) Un bloc qui représente le support physique d'un nœud I²C pour la partie écriture. Le transistor de sortie est ici représenté par un interrupteur, une résistance et une capacité.
- (b) Un réseau RLC qui représente les lignes SDA et SCL [Vas05].
- (c) Des résistances de charge du bus représentant le nœud qui y est connecté à chaque ligne.

Les signaux d'entrée du circuit (SDA IN et SCL IN) sont deux signaux numériques conformes au protocole I²C et d'une durée de 200 μ s, mais qui évoluent aléatoirement. La

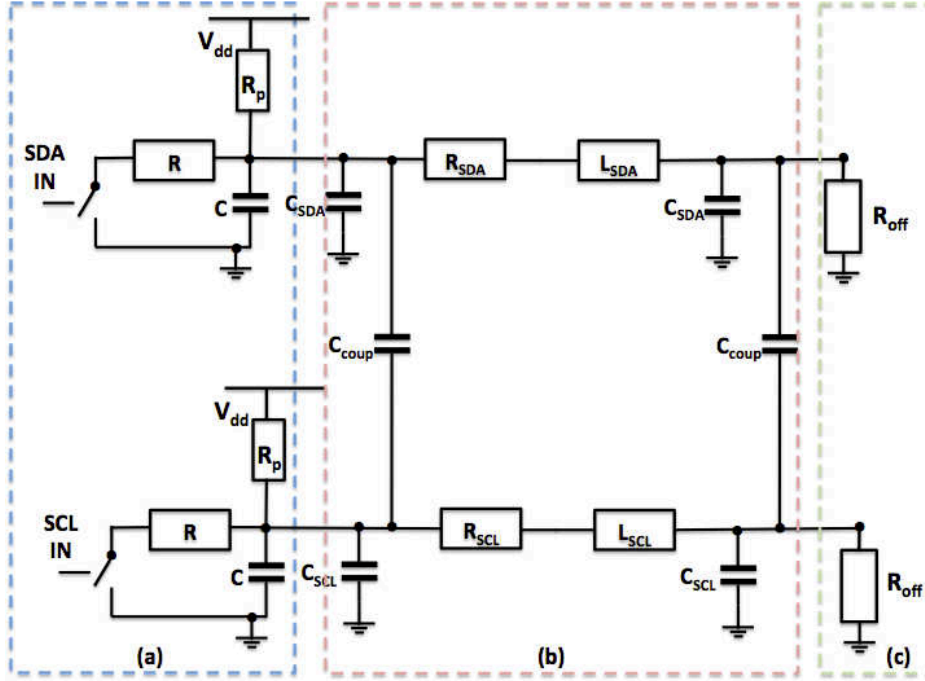


FIGURE 4.11 – Circuit équivalent utilisé dans la première étape : (a) Interface avec le contrôleur de bus (b) Lignes de bus SDA et SCL (c) Résistances de charge.

fréquence du signal varie, mais ne dépasse pas 100 kHz. En respectant la norme I²C, les temps de montée et de descente sont fixés à 7 ns. Pour faciliter l'apprentissage du réseau de neurones et lui permettre de mieux gérer le comportement des signaux SDA et SCL, nous insérons quelques valeurs supplémentaires entre les niveaux logiques "0" et "1" (cf. figure 3.22). La simulation de ce circuit avec NGSPICE, nous permet d'obtenir 200,000 couples d'entrées/sorties-désirées pour les lignes SDA et SCL.

Dans la deuxième étape, l'apprentissage du réseau de neurones pour ce système est réalisé par la *Neural Network Toolbox* de MATLAB. Pour prendre en compte l'effet de mémorisation de la tension provoquée par les capacités mutuelles du modèle de référence, nous choisissons d'utiliser le réseau NARX (*Nonlinear Autoregressive network with exogenous inputs*) [CBG90]. Ce réseau est un réseau de neurones bouclé représentable dans la *Neural Network Toolbox* de MATLAB par la fonction "newnrx". Son architecture est présentée dans la figure 4.12. Il contient une couche d'entrée, une couche cachée et une couche de sortie. Il comporte également une boucle de rétroaction des sorties du réseau vers la couche cachée. Comme présentée dans la section 3.3.3, la fonction d'activation de la couche cachée est le tan-sigmoïde (équation 3.5) et la fonction d'activation de la couche de sortie est la fonction purelin (équation 3.6). Le réseau comprend naturellement deux entrées et deux sorties qui sont les tensions des lignes SDA et SCL. Il est également nécessaire de fixer les paramètres de configuration de l'architecture du réseau NARX [DBH09] : le nombre de délais d'entrée (n_u), le nombre de délais de sortie (n_y), le nombre de neurones dans la couche cachée (S) et epoch (le nombre de mises à jour effectuées au cours de l'apprentissage). Après différentes expérimentations, nous choisissons les valeurs données dans le tableau 4.1, qui permettent d'obtenir la meilleure précision. A la fin de la deuxième étape, nous créons un fichier "busi2c.nn" qui contient les informations décrivant notre réseau NARX. Ce fichier sera utilisé dans la troisième étape.

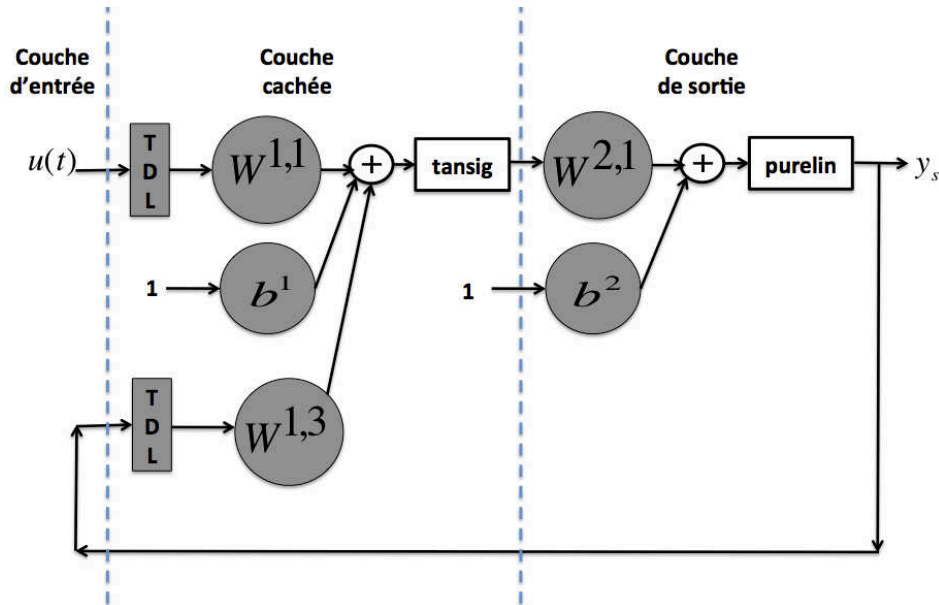


FIGURE 4.12 – Architecture du réseau NARX.

TABLE 4.1 – Paramètres de configuration de l'architecture du réseau NARX

Type de réseau	n_u	n_y	S	epoch
NARX	2	4	10	4,500

Dans la troisième étape, notre classe "neuralnetwork" (voir section 3.4.4) en C++ prend en entrée notre fichier "busi2c.nn" pour instancier le réseau de neurones dans un module SystemC-AMS TDF. Pour montrer l'intérêt de nos modèles d'intégrité du signal à haut niveau d'abstraction et, plus particulièrement, du moteur de simulation TDF, nous avons implémenté le bloc non-fonctionnel de deux façons : d'une part en utilisant le moteur de simulation TDF conventionnel et d'autre part le moteur TDF dynamique :

- TDF conventionnel : Il n'accepte qu'un pas de temps ou un délai fixe. Pour notre bloc non-fonctionnel, ce pas de temps est fixé à 1 ns, ce qui est identique au pas de simulation NGSPICE du modèle d'apprentissage. A chaque nanoseconde, on calcule donc une valeur de la sortie (cf. figure 4.13.a).
- TDF dynamique : Il permet de changer le pas de temps pendant la simulation. Nous nous intéressons particulièrement dans notre modèle aux phénomènes de diaphonie et à l'allure des transitions des signaux sur les lignes SDA et SCL. Les transitions sont donc les instants que notre modèle doit retranscrire avec le plus de précision, ce qui n'est pas le cas des zones où les signaux SDA et SCL sont stables. A l'aide du TDF dynamique, nous fixons donc le pas de temps à 1 ns quand il y a un changement d'état sur une des lignes, et à 3 ns (pour garder une précision correcte) sinon (figure 4.13.b).

Si ces deux modèles non-fonctionnels permettent de représenter les mêmes phénomènes d'intégrité du signal (puisque'ils incluent le même réseau de neurones), le modèle TDF dynamique doit permettre de diminuer le temps de simulation, puisque durant les phases

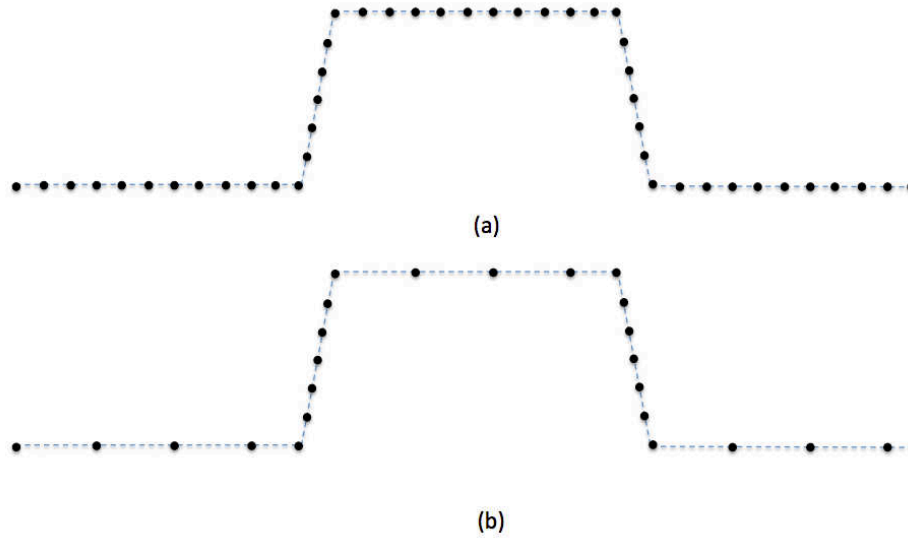


FIGURE 4.13 – Pas de temps du modèle SystemC-AMS : (a) TDF conventionnel et (b) TDF dynamique.

où les signaux sur le bus I²C sont stables, le réseau de neurones du modèle TDF dynamique sera activé moins souvent que celui du modèle TDF conventionnel.

Pour connecter le module TDF du réseau de neurones avec notre bloc fonctionnel, nous avons ensuite développé deux adaptateurs pour ce système : un module d'insertion de valeurs et un détecteur de seuil, comme nous l'avons présenté dans la chapitre 3. Tous ces modèles constituent la plate-forme illustrée dans la figure 4.8.

4.2.4 Résultats de simulation

4.2.4.1 Introduction

Dans la section précédente, nous avons présenté la construction des modèles de notre cas d'étude I²C en appliquant notre méthodologie. Nous avons notamment proposé de bâtir notre bloc non-fonctionnel de deux façons différentes, pour qu'il soit simulé avec le moteur TDF conventionnel ou le moteur TDF dynamique. Nous espérons que ces deux blocs non-fonctionnels aient des résultats de précision comparable, mais que la simulation avec le bloc basé sur le TDF dynamique soit plus rapide. Nous présentons à présent la simulation de nos modèles ainsi que leurs performances (précision, vitesse de simulation). Ces résultats seront comparés à ceux du modèle de référence.

4.2.4.2 Résultats et analyse

La figure 4.14 présente la simulation de notre seul bloc fonctionnel (cf. figure 4.10). Les signaux sont représentés sous forme numérique et l'intégrité du signal n'est donc pas prise en compte. Dans cette figure, les chronogrammes montrent les valeurs transmises sur les lignes SDA et SCL pendant l'écriture par le microcontrôleur de deux octets dans la RAM. Sur la ligne SCL, un signal d'horloge de 83 kHz est transmis. Sur la ligne SDA,

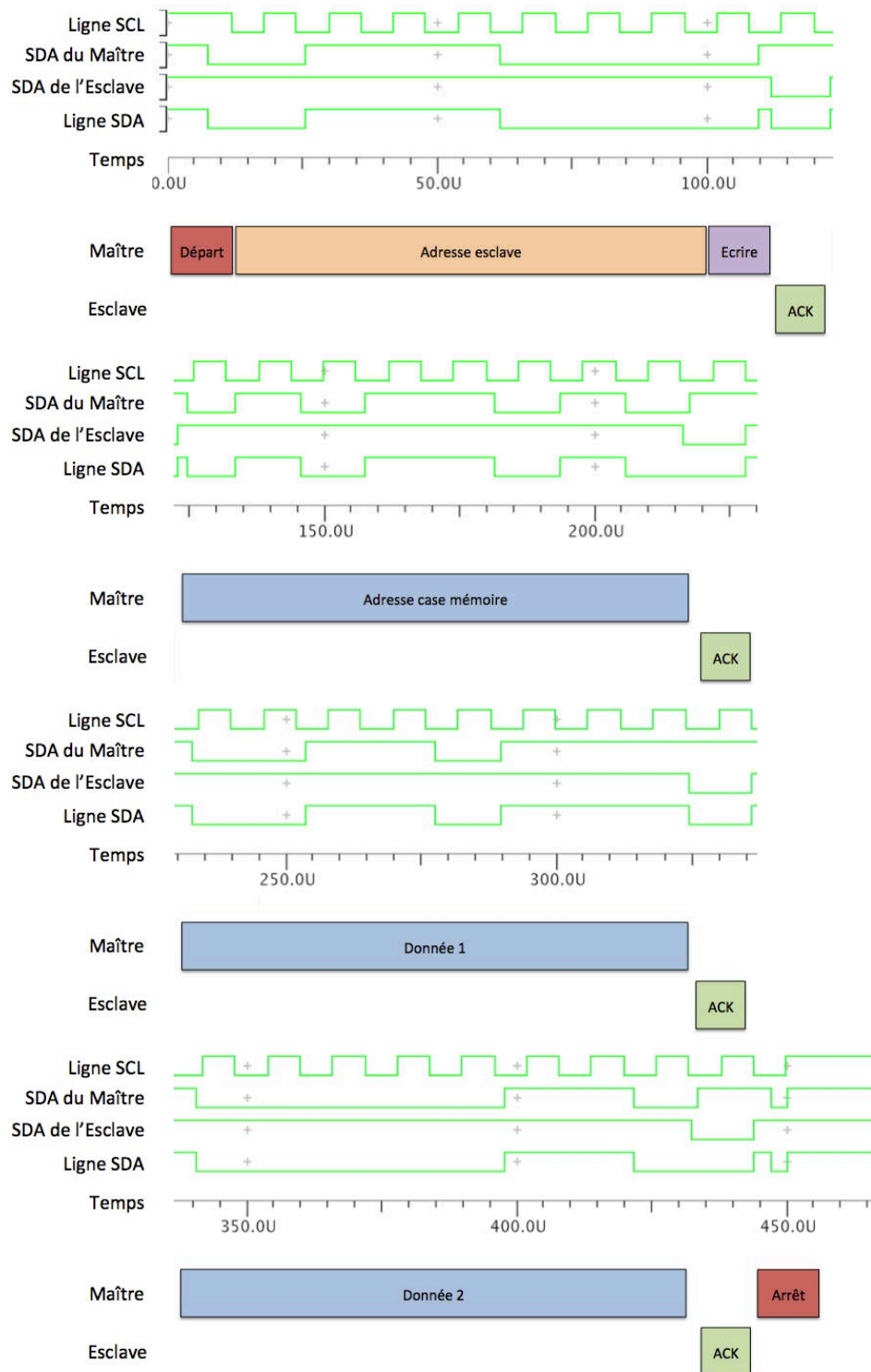


FIGURE 4.14 – Simulation du bloc fonctionnel lors d’une écriture.

la transaction commence par une condition de départ. L’octet suivant contient l’adresse de la RAM, et la commande d’écriture. L’octet suivant indique la case mémoire qui sera écrite. Enfin, la donnée sur 16 bits est transmise en 2 octets. Un bit d’acquiescement est par ailleurs envoyé par l’esclave à la fin de chaque octet. Enfin, une condition d’arrêt est

envoyée sur le bus pour terminer la transmission. Cette procédure d'écriture dure $500\ \mu\text{s}$. Pour observer plus clairement, un agrandissement sur l'intervalle $150\ \mu\text{s}$ à $190\ \mu\text{s}$ est présenté dans la figure 4.15. Ces résultats montrent que notre bloc fonctionnel permet bien de modéliser indépendamment les fonctionnalités du système.

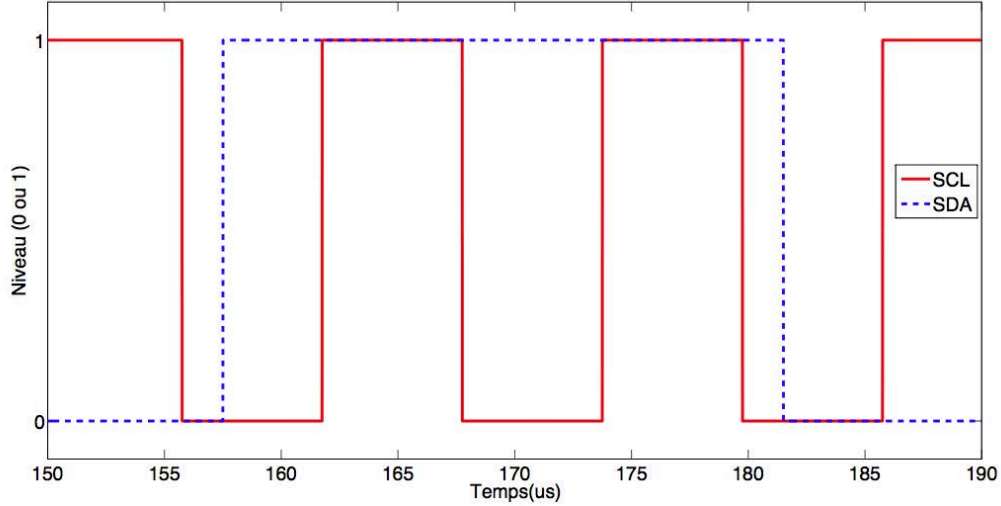


FIGURE 4.15 – Simulation du bloc fonctionnel (détail).

L'ajout des blocs non-fonctionnels permet de caractériser les performances du système en terme d'intégrité du signal. Les résultats des deux différents modèles non-fonctionnels sont présentés :

- Pour le modèle réalisé avec le moteur TDF conventionnel : Le pas de temps a été fixé à 1 ns. Nous effectuons la même transmission que précédemment, ce qui représente environ 500,000 points de simulation. La figure 4.16 présente le comportement du système dans l'intervalle de $150\ \mu\text{s}$ à $190\ \mu\text{s}$ de la simulation. Dans cette figure, les signaux sont représentés sous une forme analogique et les phénomènes de l'intégrité du signal sont pris en compte. Nous pouvons apprécier l'apport du bloc non-fonctionnel en comparant cette simulation avec la simulation idéale du bloc fonctionnel (figure 4.15). Les phénomènes dus au temps de réponse et à la diaphonie apparaissent bien dans la simulation.
- Pour le modèle réalisé avec le moteur TDF dynamique : Le pas de temps de ce modèle varie entre 1 ns et 3 ns, selon le comportement des signaux SDA/SCL. Pour simuler la même transmission présentée dans la figure 4.14, ce modèle calcule désormais environ 168,000 points. La figure 4.17 présente le comportement du système dans l'intervalle $[150\ \mu\text{s}; 190\ \mu\text{s}]$. Le comportement du modèle est similaire à celui du modèle TDF conventionnel.

Cette similarité est due au fait que les réseaux de neurones utilisés dans ces deux types de blocs non-fonctionnels sont identiques (cf. tableau 4.1). Nous avons également simulé le modèle SystemC-AMS ELN de référence et l'avons comparé à nos modèles TDF. La figure 4.18 présente la superposition des signaux issus des simulations de ces trois modèles. On peut remarquer que les phénomènes d'intégrité du signal apparaissent de façon similaire avec les trois modèles comme le montre le tableau 4.2. Les erreurs relatives absolues (*Relative absolute error*, RAE) sont de l'ordre de 3% et les coefficients de corrélation entre nos modèles et le modèle de référence sont proches de l'idéal. Les temps de simulation des

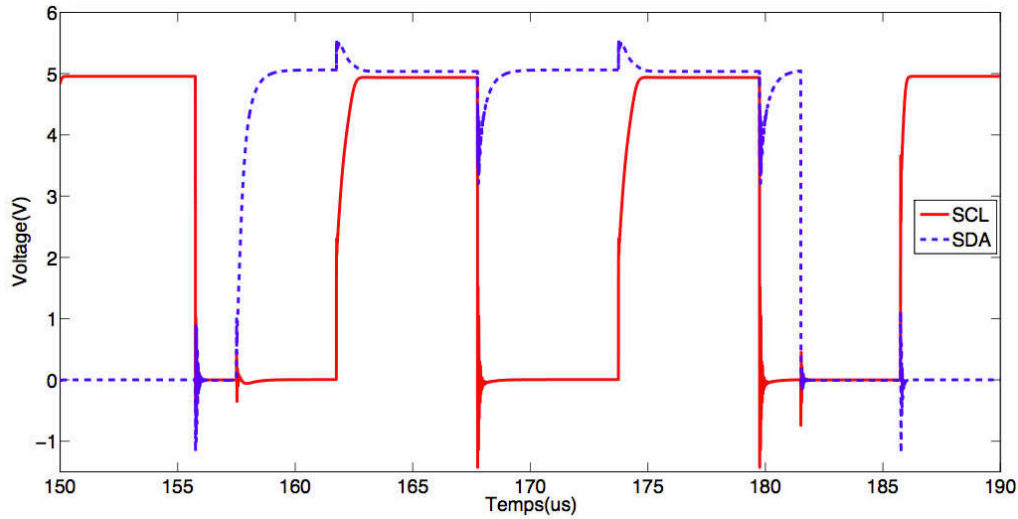


FIGURE 4.16 – Simulation du bloc non-fonctionnel avec le moteur TDF conventionnel (détail).

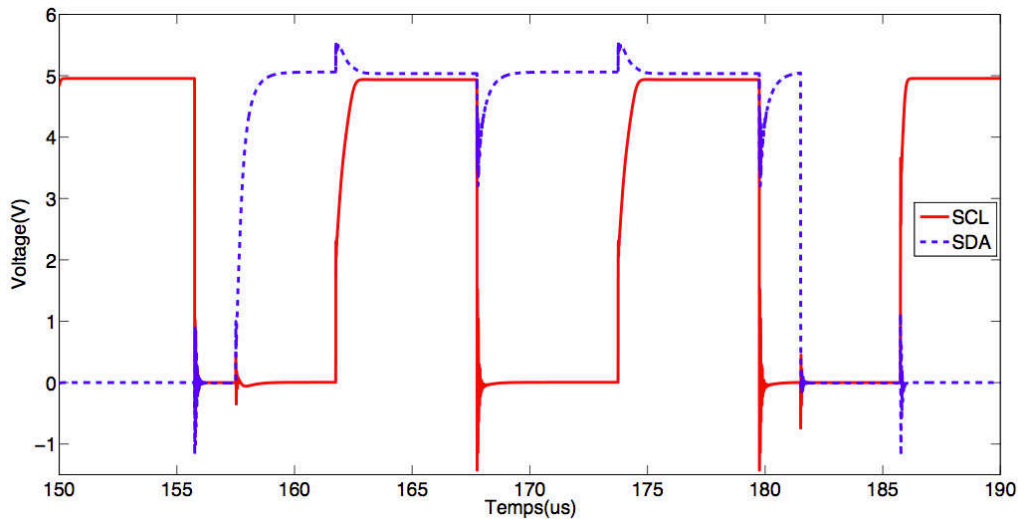


FIGURE 4.17 – Simulation du bloc non-fonctionnel avec le moteur TDF dynamique (détail).

différents modèles sont aussi donnés dans le tableau 4.2. Nous indiquons comme repère la simulation de notre bloc fonctionnel seul, afin de mesurer l'impact du modèle d'intégrité du signal sur le temps de simulation global. Nos modèles TDF (conventionnel et dynamique), ainsi que le modèle de référence ont en effet une partie fonctionnelle similaire (modèle SystemC des nœuds). La comparaison est donc pertinente. Nous constatons que le coût de nos modèles d'intégrité du signal est proportionnellement assez importante, mais que la simulation conserve une durée plus que raisonnable (5 à 15 secondes environ). La simulation de notre modèle TDF conventionnel est comparable, quoiqu'un peu plus rapide que celle du modèle de référence. Le modèle TDF dynamique permet d'accélérer sensiblement la simulation (diminution de 60% par rapport au modèle TDF conventionnel). Cela vient du fait que le bloc développé en TDF dynamique calcule moins de points (environ 33.3% des points calculés avec le modèle TDF conventionnel).

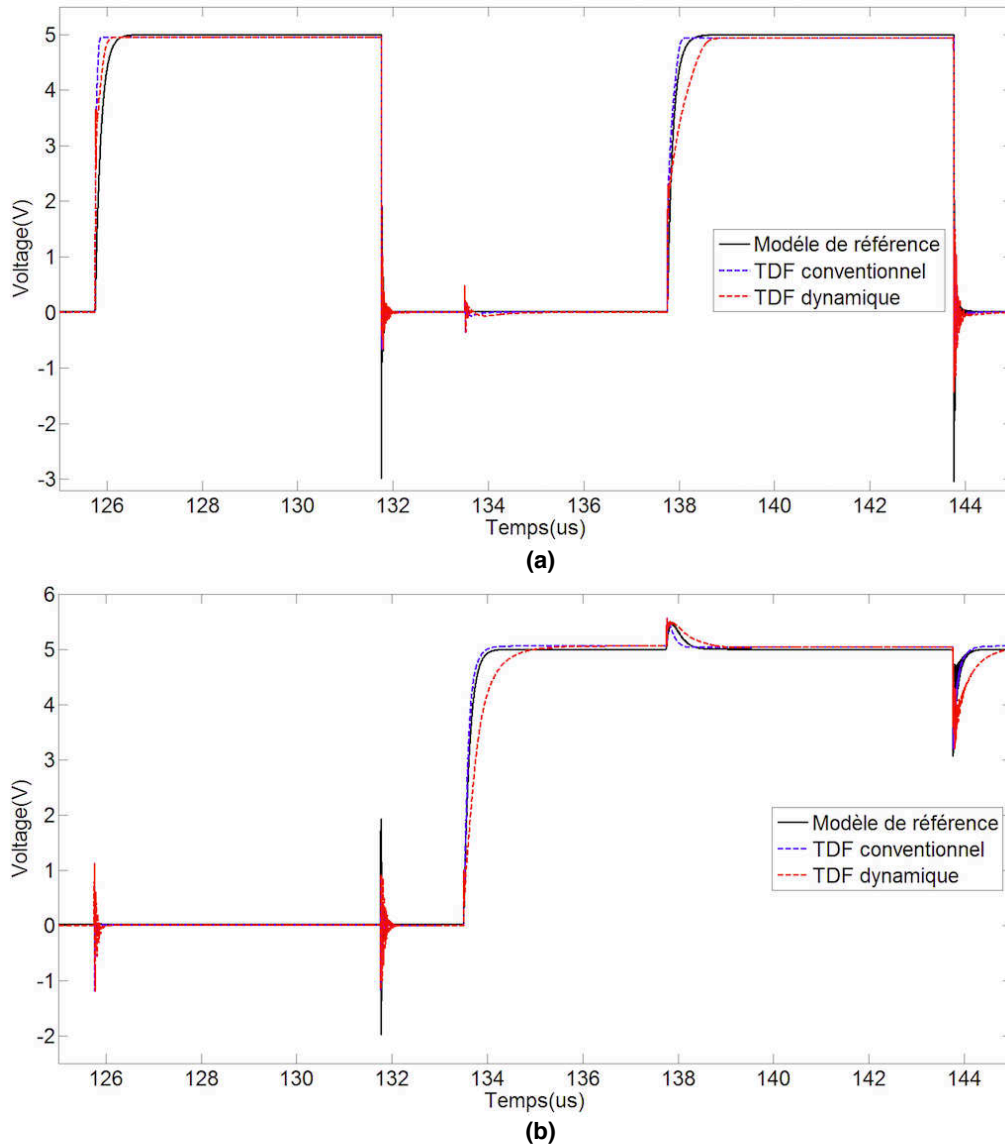


FIGURE 4.18 – Superposition des résultats de nos différents modèles et du modèle référence : (a) SCL (b) SDA.

4.2.5 Conclusion

Dans cette section, un système communiquant via un bus I²C a d'abord été présenté, puis modélisé en s'appuyant sur notre méthodologie. La simulation de notre modèle a une bonne précision et une grande vitesse de simulation, notamment le modèle basé sur le moteur TDF dynamique. Cela démontre la faisabilité de notre méthodologie.

Ce premier résultat appelle néanmoins un certain nombre de questions :

1. La vitesse de transmission du bus I²C est limitée à 5 Mbit/s en *Ultra Fast-mode*. Cependant, ce débit est bien inférieur à celui offert par d'autres protocoles de communication plus récents (comme HDMI [hdm09], DisplayPort [dis10], USB 3.0 [usb11], etc.), qui travaillent à des fréquences de l'ordre du gigahertz. Est ce que notre mé-

TABLE 4.2 – Performances de nos modèles pour le système du bus I²C

Type de modèle	Ligne de bus	RAE(%)	Coefficient de corrélation	Temps de simulation(<i>Intel Core i5-660 3.33GHz, RAM 2Gbits</i>)
Modèle idéal (bloc fonctionnel)	N/A	N/A	N/A	0.0455 s
Modèle de référence (M. Alassir)	N/A	N/A	N/A	16.386 s
TDF conventionnel	SCL	2.91	0.9952	13.190 s
	SDA	1.70	0.9991	
TDF dynamique	SCL	3.10	0.9962	5.246 s
	SDA	2.70	0.9970	

thodologie est adaptée pour de telles fréquences ?

2. Notre méthode apporte une aide au concepteur, dans le sens où celui-ci va pouvoir analyser assez tôt dans le processus de conception les éventuels problèmes d'intégrité du signal, grâce à notre méthodologie de simulation au niveau système. Est-il possible de renforcer cette aide en rendant nos modèles paramétrables ? Cela permettrait ainsi au concepteur de savoir a priori comment par exemple configurer au mieux un émetteur dans son application.
3. Nous avons utilisé un seul réseau de neurones pour modéliser l'intégrité du signal de l'ensemble de l'application, à savoir le nœud 8051, le nœud RAM et le bus I²C. Cette solution a deux inconvénients potentiels :
 - Si notre plate-forme contient 10 ou 20 nœuds au lieu de seulement 2, la complexité du réseau de neurones va augmenter et il ne sera pas aisé de faire un apprentissage efficace.
 - Si on change un nœud de la plate-forme, il faut reconstruire un réseau de neurones pour l'ensemble du système.

Pouvons-nous alors adapter notre méthode pour avoir un réseau de neurones par composant de l'application ? Cela permettrait de résoudre les problèmes de complexité et apporter une certaine modularité à notre méthodologie.

Dans la section suivante, nous répondrons à ces questions à l'aide d'une application autour d'un bus USB 3.0.

4.3 Deuxième Application : système communiquant via un bus USB 3.0

Dans la section précédente, nous avons montré la faisabilité de notre méthodologie grâce à la modélisation d'une plate-forme autour d'un bus I²C. Cependant, pour compléter

et améliorer notre méthodologie, nous avons posé à la fin de la section précédente quelques pistes d'évolution. Pour illustrer et mettre en œuvre ces évolutions, nous présentons et modélisons à présent un système communiquant via un bus USB 3.0.

Cette section est structurée en cinq parties. Dans la section 4.3.1, nous présentons le protocole USB 3.0. Dans la section 4.3.2, nous présentons le système à modéliser, ainsi que le modèle de référence. Le développement de nos modèles est décrit dans la section 4.3.3. Puis, nous montrons les résultats de simulation dans la section 4.3.4. Nous terminons cette partie par une conclusion dans la section 4.3.5.

4.3.1 Protocole USB 3.0

4.3.1.1 Généralités

La première version du protocole USB (*Universal Serial Bus*) a été publiée en 1996 par sept partenaires industriels (Compaq, DEC, IBM, Intel, Microsoft, NEC et Northern Telecom) avec pour objectif principal d'améliorer la connectivité de périphériques avec un ordinateur. La vitesse définie par l'USB 1.0 est de 1.5 Mbps. Par ailleurs, le protocole propose les concepts révolutionnaires à l'époque de connexion à chaud pour un périphérique (connecter et déconnecter sans avoir besoin de redémarrer) et de *Plug and Play* (le système reconnaît automatiquement le périphérique). En 1998, la version USB 1.1 apporte des changements dans les vitesses de communication : 1.5 Mbps en *Low Speed* (LS) et 12 Mbps en *Full Speed* (FS). En 2000, la spécification USB 2.0 pousse le débit à 480 Mbps en *High Speed* (HS). Depuis le lancement de sa première version en 1996, l'USB est devenue la technologie d'interface la plus populaire dans le marché de l'électronique grand public. La norme USB fournit en effet une interface de connexion rapide et efficace pour des appareils tels que les ordinateurs personnels, les téléphones mobiles, les claviers, les imprimantes, les appareils photo, etc. Cependant, aujourd'hui, les communications entre les différents périphériques électroniques sont devenues de plus en plus importantes. De plus, la taille des données échangées entre les appareils a énormément augmenté : par exemple, la vidéo HD (Haute Définition) entraîne des transferts de dizaines de giga-octets de données entre les différents appareils. Les vitesses des protocoles USB 1.x et 2.0 ne suffisent plus à satisfaire cette demande.

En 2008, une spécification USB 3.0 est annoncée pour une vitesse théorique de 5 Gbps grâce au bus *SuperSpeed*. USB 3.0 est rétro-compatible avec les standards USB précédents. Contrairement aux normes USB 1.x et 2.0 qui transmettent les données dans une seule direction à la fois, l'USB 3.0 est capable de lire et d'écrire des données simultanément. L'USB 3.0 est une norme *Plug and Play* rapide, capable de répondre à la demande de vitesse des applications actuelles. Les ventes mondiales des appareils compatibles avec l'USB 3.0 devraient atteindre les 3 milliards d'unités en 2018 [usb13].

Un câble USB 3.0 dispose de quatre paires de fils de connexions comme la montre la figure 4.19 : une paire de fils pour l'alimentation (V_{bus}) et la masse (GND), une paire de fils (D+ et D-) pour transférer des données au protocole USB 2.0, et deux autres paires de fils (SSTX+/SSTX- et SSRX+/SSRX-) dédiées à la liaison *SuperSpeed* USB 3.0 (une paire pour la voie d'émission et l'autre pour la voie de réception). La paire de fils USB 2.0 et les deux paires *SuperSpeed* sont des lignes qui fonctionnent en mode différentiel. Sur les deux paires de fil *SuperSpeed*, les données peuvent être transférées à des vitesses théoriques de 5 Gbps, soit 10 fois plus vite qu'USB 2.0.

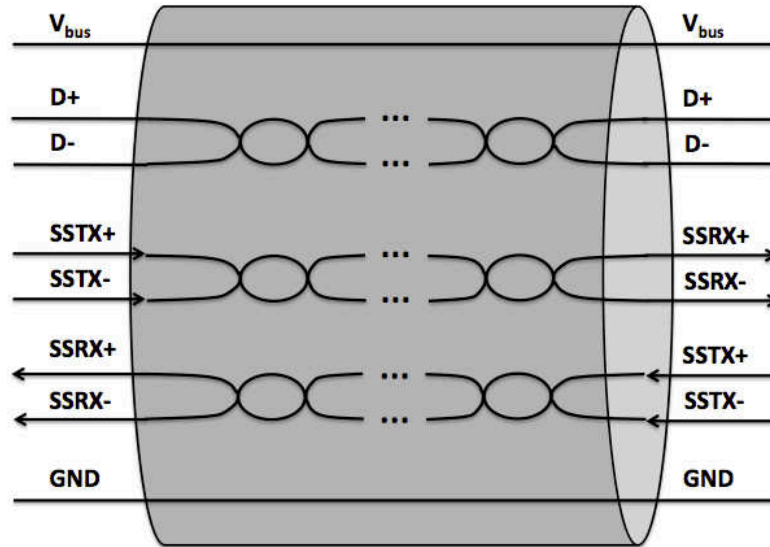


FIGURE 4.19 – Câble USB 3.0.

4.3.1.2 Architecture de la liaison SuperSpeed

Un système de transmission USB 3.0 repose donc sur un bus *SuperSpeed* associé à un bus USB 2.0. Il comporte l'hôte, les périphériques et l'interconnexion USB 3.0 (figure 4.20). L'hôte est le maître du système. Il gère toutes les activités sur le bus en initiant et ordonnant toutes les transactions. Les périphériques sont les esclaves du système. Ils répondent aux requêtes de l'hôte. L'interconnexion USB 3.0 comprend les éléments nécessaires aux périphériques pour se connecter à l'hôte. Cela inclut le câble USB 3.0 (avec le lien *SuperSpeed*), le hub USB et les connecteurs. Dans [usb11], la liaison *SuperSpeed* est présentée en trois couches (figure 4.21) :

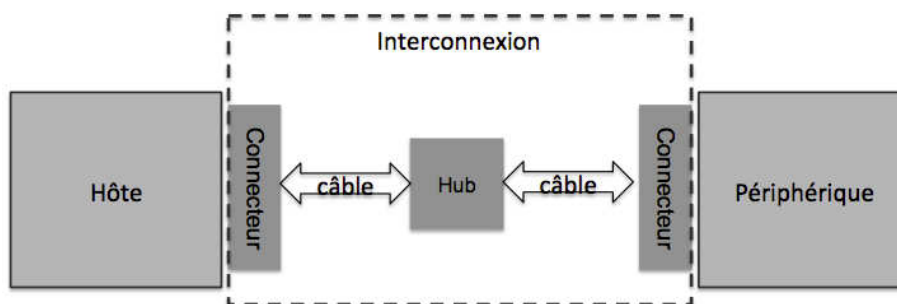
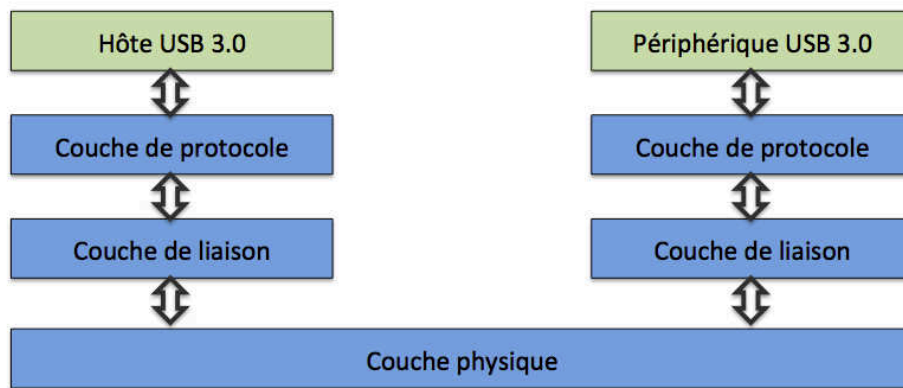


FIGURE 4.20 – Architecture d'un système USB 3.0.

- Couche physique : La couche physique définit la partie matérielle de la connexion *SuperSpeed*. Elle est représentée par les deux chemins *SuperSpeed* (émission et réception). On compte trois éléments pour chaque chemin : un émetteur (Tx), un canal, et un récepteur (Rx). Le canal représente le bus. L'émetteur et le récepteur ont des architectures similaires, mais avec des fonctionnalités duales. Un certain nombre de

FIGURE 4.21 – Architecture d’une connexion *SuperSpeed* USB 3.0.

techniques sont par ailleurs utilisées pour améliorer les performances de la liaison à haute fréquence, par exemple, la transmission différentielle, une pré-emphase du signal dans l’émetteur et une égalisation dans le récepteur.

- Couche de liaison : Un port de la liaison *SuperSpeed* comprend une partie matérielle définie par la couche physique et une partie logique définie dans la couche de liaison. Cette couche sert à définir les machines d’état qui initialisent la couche physique et gèrent son fonctionnement. De plus, cette couche distingue le type et la structure des paquets manipulés par la couche protocole (voir section suivante). Enfin, elle détecte les erreurs de transmission.
- Couche de protocole : La couche de protocole gère le flux de données entre l’hôte et les périphériques. Elle convertit les requêtes de l’hôte ou les réponses du périphérique en transactions contenant les paquets qui sont des séquences d’octets de données.

4.3.1.3 Protocole de la liaison *SuperSpeed*

Dans un lien *SuperSpeed*, l’échange des données entre l’hôte et le périphérique est réalisé sous forme de paquets. Le standard USB 3.0 propose quatre types de paquet différents :

- *Link Management Packet* (LMP) : Il gère la liaison entre les deux appareils connectés (test de la liaison, paramétrage du timeout, débit).
- *Data Packet* (DP) : Il contient les données qui sont transférées entre l’hôte et le périphérique.
- *Transaction Packet* (TP) : Il permet de gérer le flux des paquets de données entre l’hôte et les périphériques (requête de transmission, *acknowledge*, ...), et également de configurer les hôtes ou les périphériques.
- *Isochronous Timestamp Packet* (ITP) : Ce type de paquet est utilisé pour synchroniser l’hôte à tous les périphériques actifs.

Ces quatre types de paquets sont composés d’un en-tête et éventuellement d’un champ de données (*payload*). L’en-tête est composée de 20 octets avec le START et comprend les champs suivants (figure 4.22) :

- Type : Indique le type de paquet transmis (LMP, DP, TP ou ITP) en 5 bits.
- Champs spécifiques (11 octets et 3 bits) : Contient des informations spécifiques à

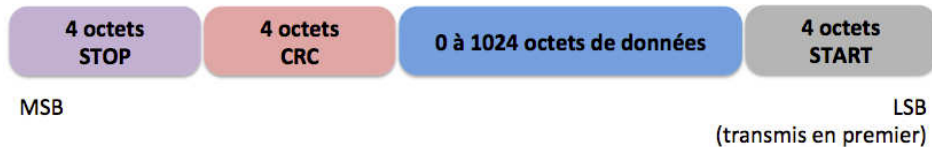


FIGURE 4.22 – Structure de l'en-tête.

chaque type de paquet. Par exemple, pour un paquet de données, c'est ici que l'on va spécifier l'adresse ou la taille du message envoyé.

- *Cyclic Redundancy Check* (CRC) (2 octets) : Permet de détecter de erreurs de transmission.
- *Link control word* (LCW) (2 octets) : Contient des informations utiles à la gestion du flux de données dans le système.

Pour les paquets de données, l'en-tête est suivi d'une *payload* (figure 4.23) composée de 4 octets de START, 0 à 1024 octets de données, un CRC de 4 octets, et 4 octets de STOP.

FIGURE 4.23 – Structure de la *payload*.

4.3.2 Présentation du système à modéliser et du modèle de référence

4.3.2.1 Système à modéliser

Comme nous l'avons présenté dans la section 4.3.1.2, une liaison USB 3.0 est typiquement structurée en trois couches. Cependant, les problèmes d'intégrité du signal apparaissent seulement au niveau de la couche physique. C'est pourquoi, nous allons nous focaliser sur la modélisation de la couche physique d'une connexion *SuperSpeed*.

La couche physique contient deux chemins *SuperSpeed* duaux : un chemin pour l'émission et un autre pour la réception. Seule la direction des données diffère. Aussi, pour démontrer la faisabilité de notre méthode pour des hautes fréquences de transmission, nous n'allons modéliser qu'un seul chemin *SuperSpeed*. Cette couche physique contient trois éléments comme l'illustre la figure 4.24 :

- Une paire de fils *SuperSpeed* : Comme dit précédemment, la transmission sur le bus *SuperSpeed* se fait à l'aide d'une paire différentielle. Les signaux ont ainsi une bonne immunité aux bruits parasites de l'environnement, des périphériques ou du câble. On suppose en effet que la perturbation va dégrader le signal de manière similaire sur chaque ligne de transmission, ce qui va limiter les effets de diaphonie pour le décodage des données. En revanche, l'interférence inter-symbole (ISI) est un problème critique pour un signal à haute fréquence. L'ISI est une distorsion du signal où le symbole transmis précédemment affecte le symbole suivant, ce qui rend la communication moins fiable. Par exemple, la figure 4.25 représente les signaux

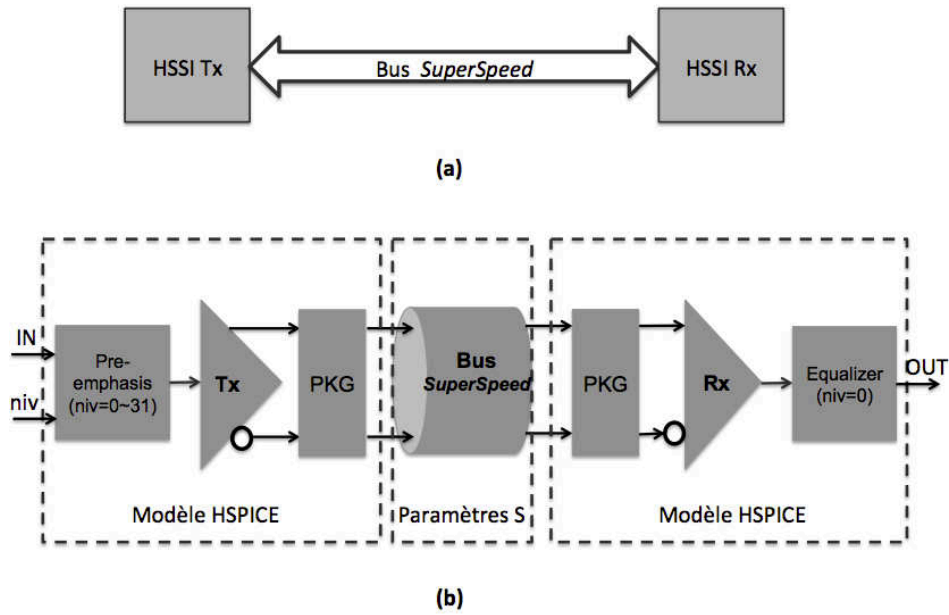


FIGURE 4.24 – (a) Système USB 3.0 à modéliser (b) Modèle de référence.

reçus dans deux cas distincts. Dans cette figure, les messages émis sont de forme numérique et sont reconstitués en plaçant un comparateur à seuil à leur arrivée. Dans le cas de la figure 4.25.a, le débit est trop élevé pour que le comparateur puisse récupérer le message. Dans le cas de la figure 4.25.b, le message est récupéré sans erreur.

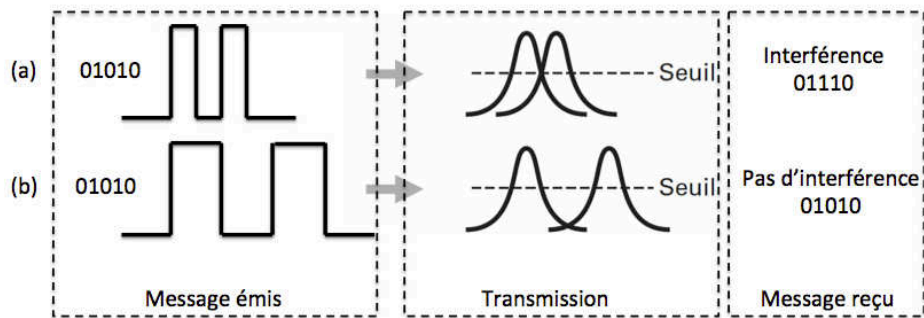


FIGURE 4.25 – Illustration de l'ISI : (a) interférences et (b) pas d'interférences.

- Un émetteur HSSI (*High Speed Serial Interface*) pour un FPGA Stratix IV GX d'Altera [Alt12] : Une fonction de pré-emphase est incluse dans cet émetteur. Le but de cette fonction est d'appliquer au signal original une déformation (à l'aide d'opération de délais ou d'inversions) afin de compenser certains effets parasites tels que l'ISI. Pour optimiser cette compensation, il faut combiner différents délais, poids et polarités. Dans l'émetteur HSSI d'Altera, la procédure de pré-emphase est mise en œuvre par un filtre à réponse impulsionnelle finie (FIR) travaillant sur les 4 derniers échantillons : le signal différentiel original (V_{OD}), l'échantillon précédent (*Pre-tap*), l'échantillon retardé d'une période (*1st Post-tap*), l'échantillon retardé de

deux périodes (*2nd Post-tap*) [Alt10]. La figure 4.26 présente l'architecture de ce filtre FIR. La sortie de ce filtre dépend des coefficients appliqués aux valeurs des quatre échantillons.

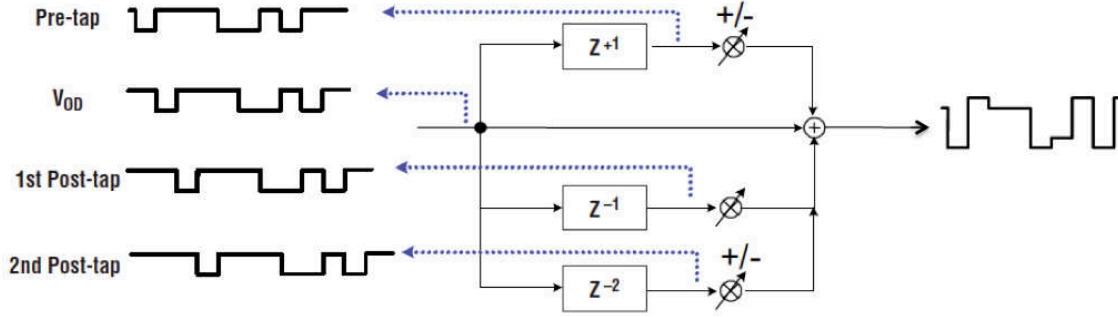


FIGURE 4.26 – Filtre FIR de l'émetteur HSSI.

- Un récepteur HSSI pour un FPGA Stratix IV GX d'Altera [Alt12] : Le récepteur HSSI propose une égalisation qui permet de compenser les effets de l'ISI après le passage du signal dans la ligne de transmission. L'égalisation agit comme un filtre passe-bande.

4.3.2.2 Modèle de référence

Nous avons présenté le deuxième système à modéliser. Nous présentons maintenant le modèle de référence de ce système utilisé pour valider notre méthodologie. Ce modèle de référence sera utilisé dans l'étape d'acquisition des couples d'entrées/sorties-désirées pour construire le réseau de neurones, et aussi dans l'étape de validation de nos résultats. Dans ce modèle de référence, le bus *SuperSpeed* est représenté par un modèle des paramètres S. Nous considérons une longueur de câble de 3 mètres. Les interfaces de l'émetteur et du récepteur sont représentées par un modèle HSPICE proposé par Altera [Alt09]. Ce modèle d'Altera modélise non seulement les interfaces, mais aussi les fonctions de pré-emphase/égalisation et les boîtiers (PKG) des composants. La figure 4.24.b montre le modèle de référence de notre système. Pour fixer les paramètres des modèles HSPICE de l'émetteur ou du récepteur, comme l'impédance de terminaison ou le couplage capacitif, il faut suivre les recommandations de la norme USB 3.0 [usb11].

Les modèles HSPICE de l'émetteur et du récepteur permettent aux concepteurs de tester, puis de valider différentes configurations de pré-emphase de l'émetteur (32 niveaux de configuration pour le *1st Post-tap*, 16 niveaux pour le *2nd Post-tap*, 16 niveaux pour le *Pre-tap*, soit $32 \times 16 \times 16 = 8,192$ configurations possibles) et d'une égalisation du récepteur (16 niveaux). Dans nos travaux, nous allons uniquement considérer la fonction de pré-emphase et travailler seulement avec le *1st Post-tap* pour démontrer que notre méthodologie peut être utilisée pour aider un concepteur à paramétrer un système de transmission USB 3.0. La figure 4.27 [Alt10] présente le processus de pré-emphase avec le *1st Post-tap*. On considère le signal original $V_p(T)$ et son inverse $V_n(T)$ aux instants T et $T - 1$. Le signal modifié par la pré-emphase est alors donné par l'équation 4.2 :

$$V_{\text{signal-pre-emphase}} = V_p(T) + x \times V_p(T - 1) - V_n(T) + x \times V_n(T - 1). \quad (4.2)$$

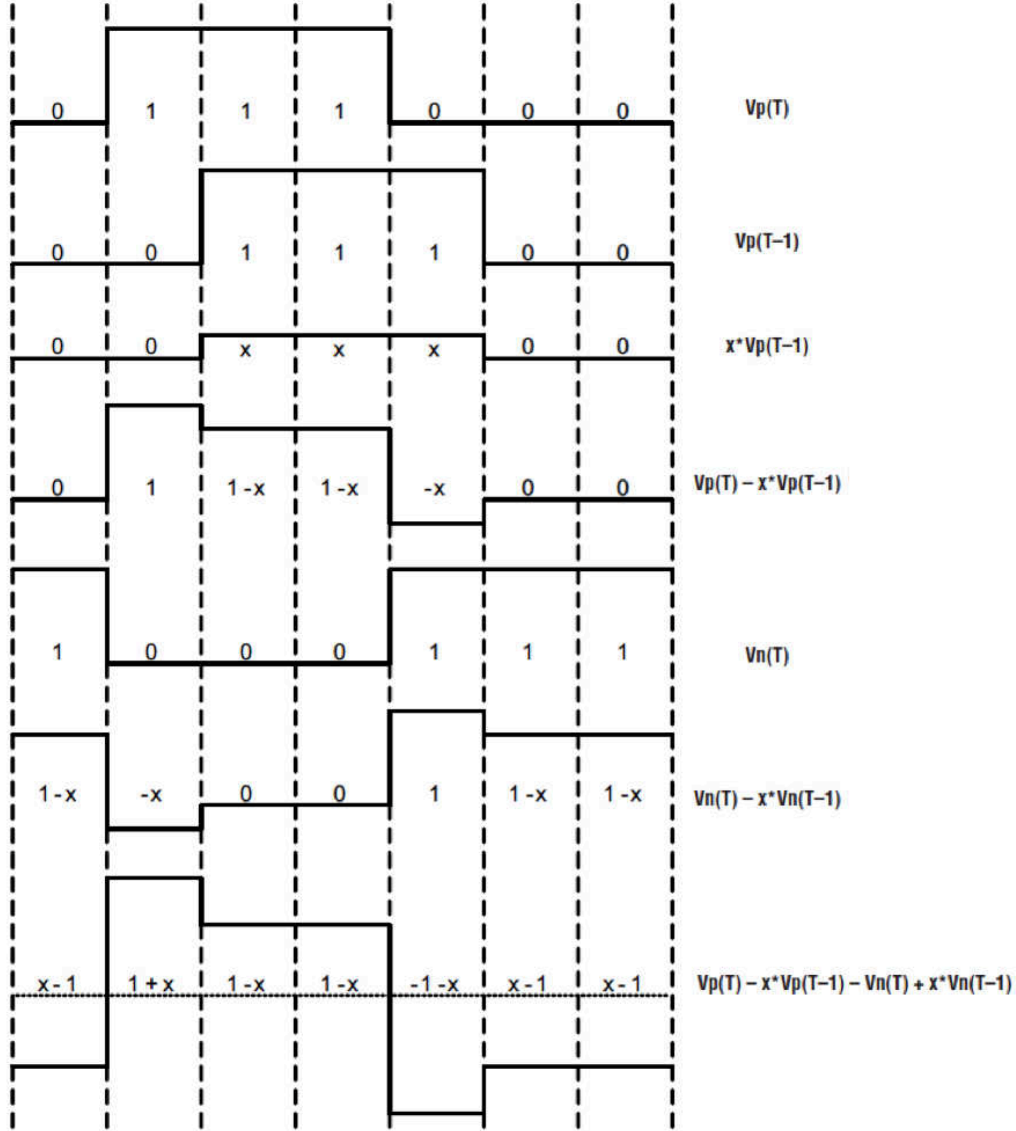


FIGURE 4.27 – Construction du signal de pré-emphase avec le 1st Post-tap.

Dans cette équation, x indique le poids de déformation (dont la valeur est fixée par le niveau de configuration). Pour illustrer cela, la figure 4.28 propose une simulation montrant l'influence des niveaux de configuration sur l'allure du signal transmis. Il est à noter que selon l'application (caractéristiques électriques de l'émetteur/récepteur, débit souhaité) la valeur du poids pour un même niveau de configuration n'est pas constante. Altera recommande de s'appuyer sur des mesures ou des simulations pour connaître la valeur de x [Alt14].

Ce modèle de référence en HSPICE est un modèle à bas niveau d'abstraction dont les performances en terme de durée de simulation sont très limitées. Ainsi, pour simuler la transmission de 200 bits, il faut compter environ 45 minutes. Il n'est donc pas envisageable de tester les 8,192 configurations de la fonction de pré-emphase de l'émetteur. Seul un modèle de plus haut niveau pourrait répondre à cette demande.

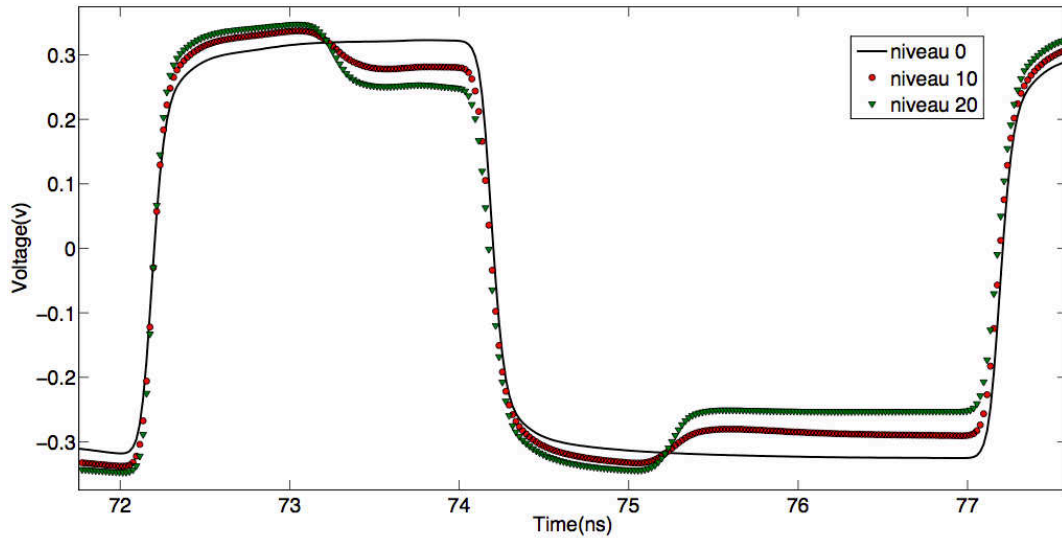


FIGURE 4.28 – Signal issu de la fonction de pré-emphase pour différents niveaux de configuration du *1st Post-tap*.

4.3.3 Développement de notre modèle

4.3.3.1 Introduction

Notre deuxième application (cf. figure 4.24) est donc la couche physique d'un système de transmission USB 3.0. Comme nous l'avons présenté dans la section 4.3.1.2, c'est à ce niveau qu'apparaissent les problèmes d'intégrité du signal. Les couches de plus haut niveau définissent la partie fonctionnelle de la transmission et correspondent donc aux blocs fonctionnels de notre méthodologie. La couche physique correspond alors aux blocs non-fonctionnels. Nous avons choisi ici de modéliser cette plate-forme seulement avec un bloc non-fonctionnel. L'architecture du bloc non-fonctionnel représentant la couche physique USB 3.0 est donnée dans la figure 4.29. Pour répondre aux trois points soulevés à la fin de la section 4.2.5, nous avons proposé et réalisé trois différents types de bloc non-fonctionnels. Nous détaillons dans cette section la construction de ces trois blocs non-fonctionnels pour ce deuxième cas d'étude.

4.3.3.2 Bloc non-fonctionnel

Dans notre méthodologie, le bloc non-fonctionnel permet de modéliser les performances d'intégrité du signal. Comme nous l'avons présenté pour le système basé sur I²C, trois étapes sont nécessaires pour construire le modèle non-fonctionnel. Pour améliorer et compléter notre méthodologie, nous allons valider trois points spécifiques :

1. Est ce que notre méthodologie peut être utilisée pour des applications à haute fréquence ?
2. Est ce que le bloc non-fonctionnel peut être utilisé pour aider un concepteur à paramétrer son système ?
3. Est-il possible d'appliquer un réseau de neurones par bloc de l'application ?

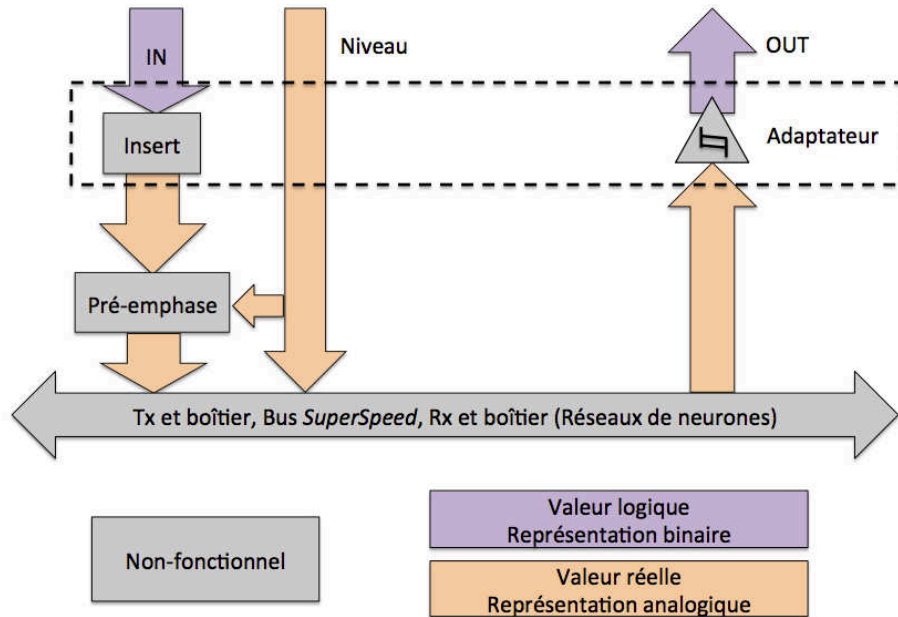


FIGURE 4.29 – Architecture du bloc non-fonctionnel représentant la liaison USB 3.0 *SuperSpeed* (Insert : Insertion).

Pour répondre à ces trois questions, nous avons proposé et réalisé trois différents types de blocs non-fonctionnels :

- **Bloc mono-réseau** : Ce bloc non-fonctionnel contient un réseau de neurones qui donne les caractéristiques d'intégrité du signal de l'ensemble des blocs de la plateforme (émetteur+boîtier, bus *SuperSpeed*, récepteur+boîtier). Pour ce modèle, le niveau de pré-emphase n'est pas paramétrable (il est fixé à 0). Ce type de bloc non-fonctionnel est réalisé pour démontrer la faisabilité de notre méthodologie à haute fréquence.
- **Bloc mono-réseau paramétrable** : Ce bloc non-fonctionnel contient un seul réseau de neurones similaire à celui du mono-réseau, mais nous pouvons dans ce cas également paramétrer le niveau de pré-emphase. Notre modèle permet de tester 5 configurations différentes (0/5/10/15/20). Ce type de bloc va permettre de répondre à la deuxième question.
- **Bloc multi-réseaux** : Ce bloc non-fonctionnel est composé de trois réseaux de neurones : un réseau de neurones pour caractériser l'émetteur et son boîtier ; un réseau pour le câble *SuperSpeed* ; un réseau pour le récepteur et son boîtier. A l'aide de ce bloc non-fonctionnel, nous répondrons à la troisième question.

Dans cette section, nous détaillons la réalisation de ces trois types de bloc non-fonctionnel, étape par étape.

Etape 1 - Acquisition des couples d'entrées/sorties-désirées

Les couples d'entrées/sorties-désirées sont obtenus en simulant le modèle de référence présenté dans la section 4.3.2.2. Le signal à transmettre est une trame numérique générée aléatoirement à un débit de 1 Gbps. Une valeur de pré-emphase (variable ou pas selon les cas) est également fournie.

- **Bloc mono-réseau** : Les couples d'entrées/sorties-désirées sont composés de la

tension d'entrée de l'émetteur (V_{IN}) pour l'entrée et de la tension de sortie du récepteur (V_{OUT}) pour la sortie-désirée (voir figure 4.30). Le niveau de pré-emphase est fixé à 0. On recueille ainsi 36,000 couples de valeurs (pour une durée simulée de 180 ns et un pas de simulation de 0.005 ns).

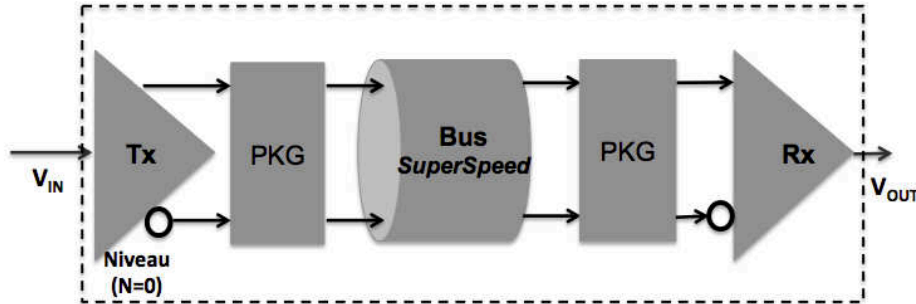


FIGURE 4.30 – Signaux utilisés dans le bloc mono-réseau.

- **Bloc mono-réseau paramétrable** : Les couples d'entrées/sorties-désirées sont composés de : la tension V_{IN} et le niveau de pré-emphase N pour l'entrée, la tension V_{OUT} pour la sortie-désirée (voir figure 4.31). Pour chaque niveau de pré-emphase pris en compte par notre modèle (5 dans notre cas), nous obtenons 9,000 couples (pour une durée simulée de 180 ns et un pas de simulation de 0.02 ns). Nous gardons moins de valeurs par niveau que pour le bloc mono-réseau afin de ne pas trop alourdir la phase d'apprentissage par un volume de données trop important.

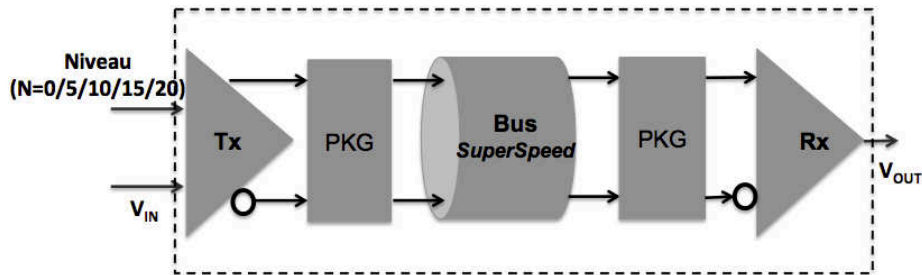


FIGURE 4.31 – Signaux utilisés dans le bloc mono-réseau paramétrable.

- **Bloc multi-réseaux** : Voici les couples d'entrées/sorties-désirées pour les trois réseaux de neurones du bloc multi-réseaux (figure 4.32) :
 1. Pour le réseau caractérisant l'émetteur et son boîtier, l'entrée est la tension d'entrée de l'émetteur (V_{IN}), les sorties-désirées sont les tensions sur la paire différentielle (V_{Tx+} et V_{Tx-}) connectée au bus SuperSpeed (figure 4.32.a).
 2. Pour le réseau caractérisant le bus, les entrées sont les tensions émises par l'émetteur sur la paire différentielle (V_{Tx+} et V_{Tx-}), les sorties-désirées sont les tensions V_{Rx+} et V_{Rx-} qui sont vues par le récepteur (figure 4.32.b).
 3. Pour le réseau caractérisant le récepteur et son boîtier, les entrées sont les tensions émises par le bus SuperSpeed sur la paire différentielle (V_{Rx+} et V_{Rx-}), la sortie-désirée est la tension de sortie V_{OUT} du récepteur (figure 4.32.c).

Chaque ensemble de valeurs est stocké dans un fichier différent afin d'être exploité dans l'étape 2. Chaque fichier contient 36,000 couples de valeurs (comme pour le bloc mono-réseau) .

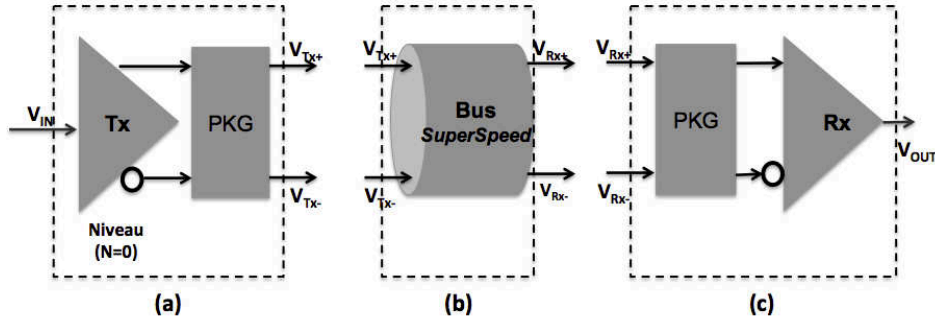


FIGURE 4.32 – Signaux utilisés dans le bloc multi-réseaux.

Etape 2 - Apprentissage du réseau de neurones

Dans cette étape, nous construisons les réseaux de neurones de nos blocs non-fonctionnels. Les réseaux de neurones utilisés pour les trois types de blocs appartiennent à la même classe d'architecture, à savoir la classe FTDNN (*Focused Time-Delay Neural Network*) [LWH90]. FTDNN est un réseau multicouches non-bouclé avec délais (figure 4.33). La *Neural Network Toolbox* de MATLAB propose une fonction "newfftd" pour créer ce réseau. Comme le réseau NARX utilisé pour notre plate-forme I²C, il contient une couche d'entrée, une couche cachée et une couche de sortie. Il permet également d'insérer des unités de retard sur les entrées. Pour les trois types de bloc non-fonctionnel, la fonction d'activation des couches cachées est le tan-sigmoïde, celle des couches de sortie est purelin. En fonction du type de bloc non-fonctionnel que nous voulons réaliser (mono-réseau, mono-réseau paramétrable ou multi-réseaux), il nous faut configurer spécifiquement le nombre de délais d'entrées (n_u), le nombre de neurones dans la couche cachée (S) et epoch. Nous présentons dans le tableau 4.3 les valeurs de ces paramètres obtenues à l'issue de la phase d'apprentissage. A la fin de cette étape, les fichiers contenant les informations de ces réseaux sont créés.

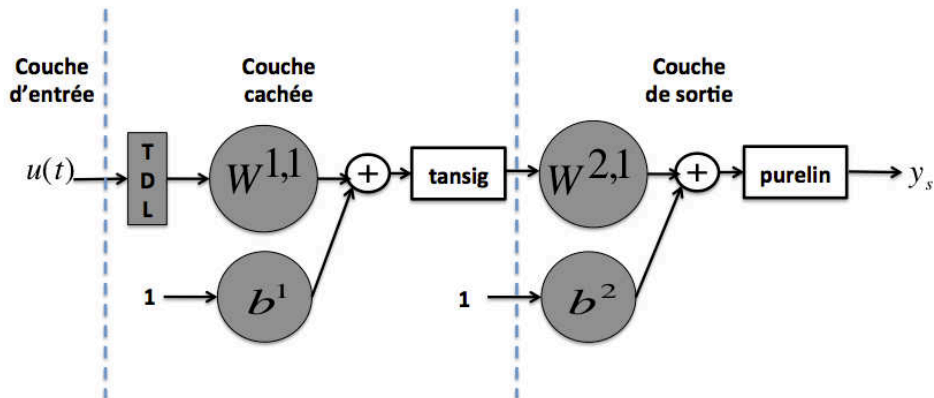


FIGURE 4.33 – Architecture du réseau FTDNN.

Etape 3 - Construction du modèle SystemC-AMS TDF implémentant le réseau de neurones et les adaptateurs

TABLE 4.3 – Paramètres de configuration des réseaux FTDNN pour les trois blocs non-fonctionnels.

Type de bloc	Sous-bloc	n_u	S	epoch
Mono-réseau	N/A	500	10	3,000
Mono-réseau paramétrable	N/A	50	10	3,000
Multi-réseaux	Emetteur	200	10	2,000
	Bus <i>SuperSpeed</i>	500	10	4,000
	Récepteur	30	10	4,000

Dans la troisième étape, la classe "neuralnetwork" nous permet à partir du fichier ".nn", de reconstruire les réseaux de neurones et de les implémenter dans un module SystemC-AMS TDF. Des adaptateurs y sont associés pour permettre de s'interfacer avec un modèle fonctionnel de l'application. Cela nous permettra d'estimer plus justement l'impact de nos modèles non-fonctionnels sur les performances de la simulation.

- **Bloc mono-réseau** : L'architecture de ce bloc non-fonctionnel est donnée dans la figure 4.34. Les adaptateurs sont le module d'insertion de valeurs et le détecteur de seuil, présentés dans le chapitre 3.

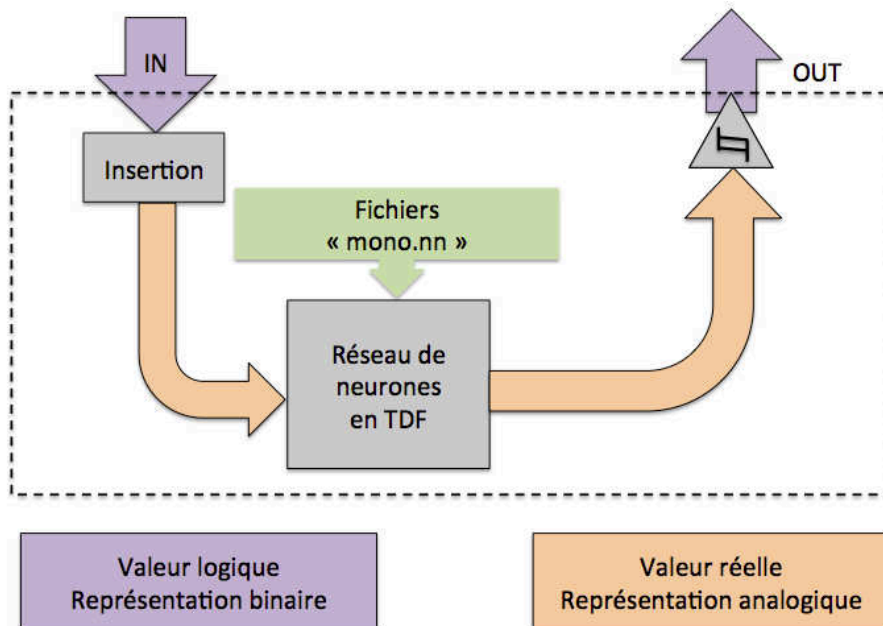


FIGURE 4.34 – Architecture du bloc non-fonctionnel mono-réseau.

- **Bloc mono-réseau paramétrable** : L'architecture de ce bloc non-fonctionnel est donnée dans la figure 4.35. Les adaptateurs sont le module d'insertion de valeurs et le détecteur de seuil présentés dans le chapitre 3. Il faut également insérer un bloc

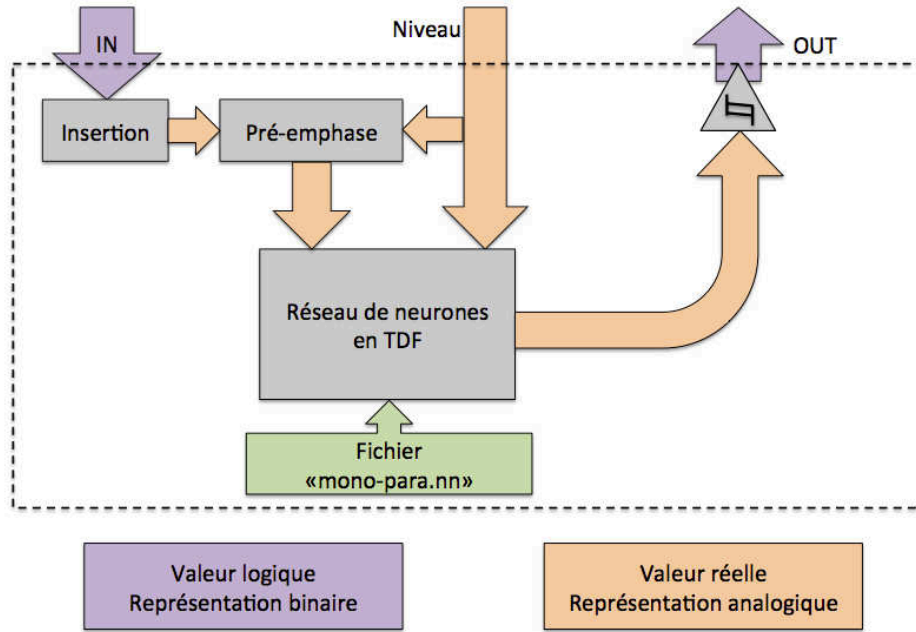
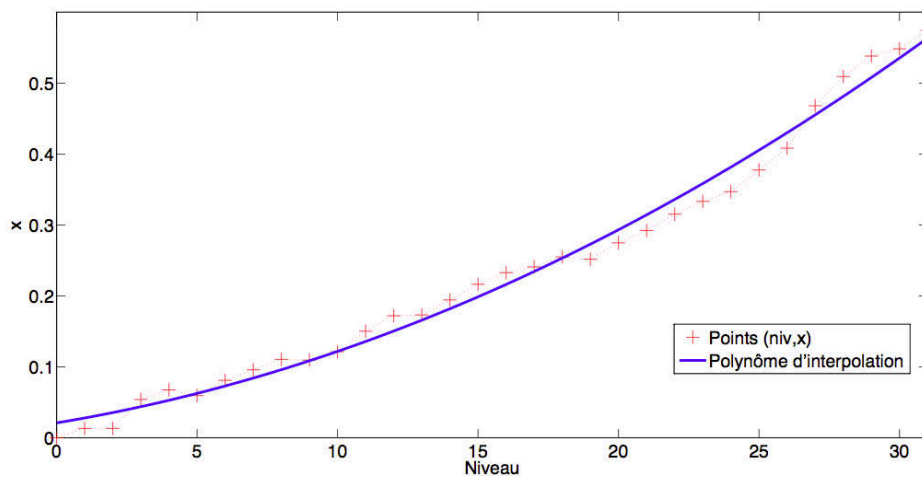


FIGURE 4.35 – Architecture du bloc mono-réseau paramétrable.

qui représente la fonction de pré-emphase. Pour cela, nous implémentons l'équation 4.2 en SystemC-AMS. Nous pouvons obtenir les signaux $V_p(T-1)$ et $V_n(T-1)$ à partir des signaux originaux ($V_p(T)$ et $V_n(T)$). À l'aide de notre modèle de référence, nous avons mesuré les valeurs de x de l'équation 4.2 pour notre plate-forme. Cela nous a permis d'obtenir la courbe de x en fonction du niveau de configuration (figure 4.36). Le polynôme d'interpolation sur les données nous donne une relation entre la valeur de x et le niveau de configuration niv de la pré-emphase (équation 4.3).

$$x = 3.53 \times 10^{-4} \times niv^2 + 6.55 \times 10^{-3} \times niv + 2.1 \times 10^{-2}. \quad (4.3)$$

Cette équation est implémentée en SystemC-AMS pour nous permettre d'obtenir

FIGURE 4.36 – Relation entre niveau de pré-emphase et x .

la fonction de pré-emphase.

- **Bloc multi-réseaux** : L'architecture du bloc multi-réseaux est donnée dans la figure 4.37. Pour reconstituer la liaison USB 3.0 (émetteur->bus *SuperSpeed*->récepteur), nous connectons la sortie d'un réseau de neurones à l'entrée du suivant. Les adaptateurs sont les mêmes que pour le bloc mono-réseau.

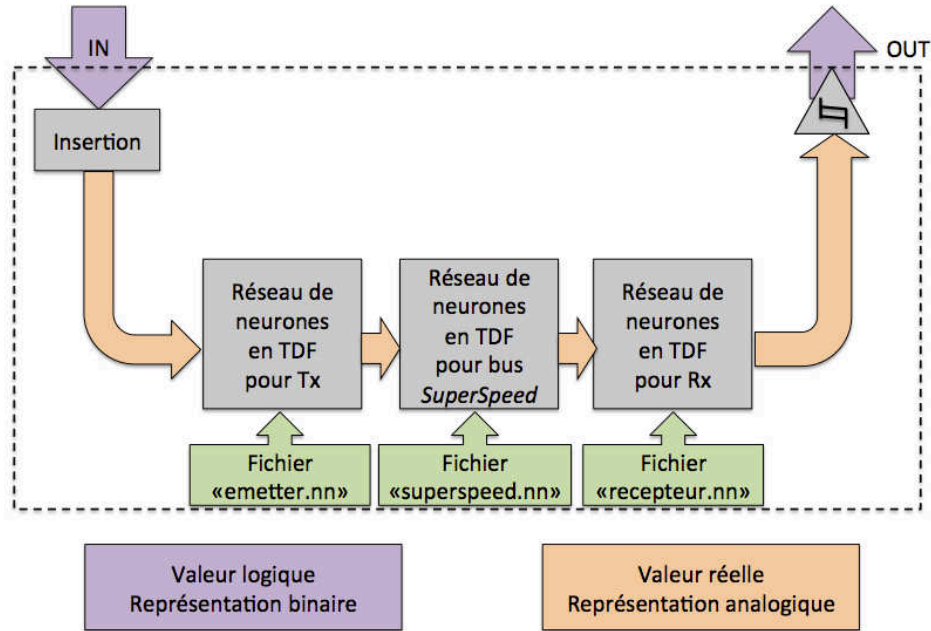


FIGURE 4.37 – Architecture du bloc multi-réseaux.

4.3.4 Résultats de simulation

4.3.4.1 Introduction

Dans la section précédente, nous avons présenté la construction de notre modèle de plate-forme USB 3.0. Nous détaillons et analysons à présent les résultats et les temps de simulation de nos trois différents modèles et les comparons avec le modèle de référence pour évaluer la pertinence de notre méthodologie.

4.3.4.2 Résultats et analyse

Tous les résultats suivants ont été obtenus en simulant une même séquence de 200 bits générée aléatoirement. Nous présentons d'abord les résultats de simulation de nos trois types de blocs non-fonctionnels et les comparons avec la simulation du modèle de référence.

Nous commençons par le bloc mono-réseau. Dans la figure 4.38, nous superposons l'allure du modèle de référence avec celle de notre modèle. L'erreur est donnée dans le tableau 4.4. Nous constatons que notre bloc mono-réseau modélise la plate-forme USB 3.0 avec précision. Cela démontre la possibilité d'appliquer notre méthodologie à des hautes fréquences de transmission, ce qui répond à la première question soulevée à la fin de la section 4.2.

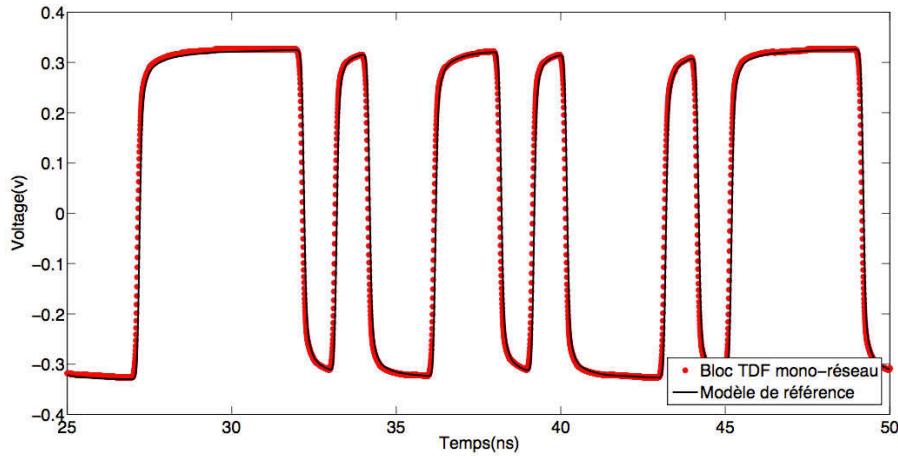


FIGURE 4.38 – Résultats de simulation du bloc mono-réseau.

TABLE 4.4 – Précision du bloc mono-réseau

	RAE (%)	Coefficient de corrélation
Bloc mono-réseau	1.07	0.9999

Nous présentons ensuite dans la figure 4.39 les résultats du modèle mono-réseau paramétrable. La figure superpose l'allure du modèle TDF et du modèle de référence avec un niveau de pré-emphase fixé à 20. Nous pouvons observer les déformations des signaux introduites par la pré-emphase, ainsi que la bonne corrélation entre les deux modèles. Cette corrélation (pour le niveau 20 et les autres niveaux) est quantifiée dans le tableau 4.5. Nous montrons ainsi que nous pouvons paramétrer la plate-forme USB 3.0 à l'aide de notre méthodologie sans affecter les performances du modèle. Il n'est pas non plus nécessaire de reconstruire un réseau de neurones à chaque fois que l'on veut tester un niveau de pré-emphase. Ceci répond à la deuxième question.

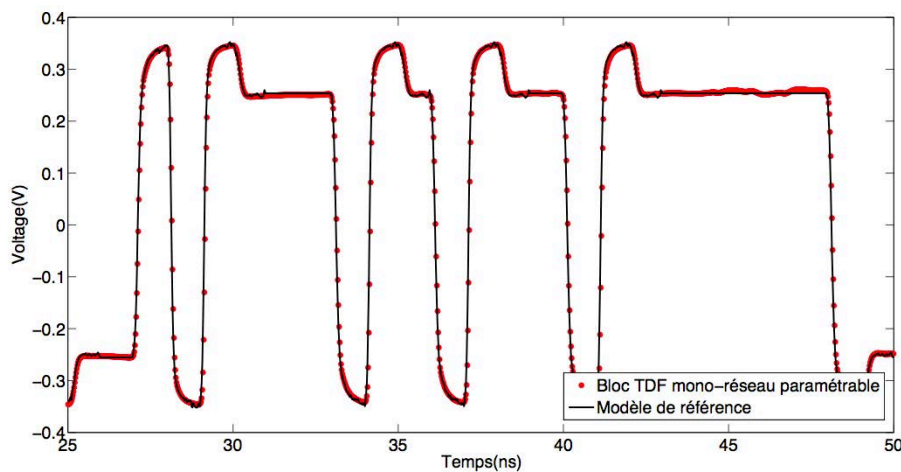


FIGURE 4.39 – Résultats de simulation du bloc mono-réseau paramétrable (niveau de pré-emphase : 20).

TABLE 4.5 – Précision du bloc mono-réseau paramétrable

Niveau de pré-emphase	RAE (%)	Coefficient de corrélation
0	2.07	0.9997
5	3.45	0.9995
10	2.09	0.9997
15	2.87	0.9998
20	1.63	0.9998

Pour le modèle multi-réseaux, nous commençons par donner les résultats des trois réseaux élémentaires. La figure 4.40 montre les résultats du réseau de neurones pour l'émetteur et son boîtier. La figure 4.41 montre les résultats du réseau de neurones pour le bus *SuperSpeed*. La figure 4.42 montre les résultats du réseau de neurones pour le récepteur et son boîtier. Puis, les résultats du modèle multi-réseaux complet (le bloc non-fonctionnel contenant les trois réseaux) sont présentés dans la figure 4.43. La précision des modèles est donnée dans le tableau 4.6. On constate que la flexibilité propre aux réseaux de neurones permet de corriger, dans une certaine mesure, les erreurs des étages précédents au lieu de les amplifier. Ainsi, la plus grande imprécision des modèles de l'émetteur est corrigée par les réseaux de neurones du bus, puis du récepteur. Nous montrons ainsi la faisabilité de notre méthode en combinant plusieurs réseaux de neurones de faible complexité dans un même modèle, ce qui répond à la troisième question.

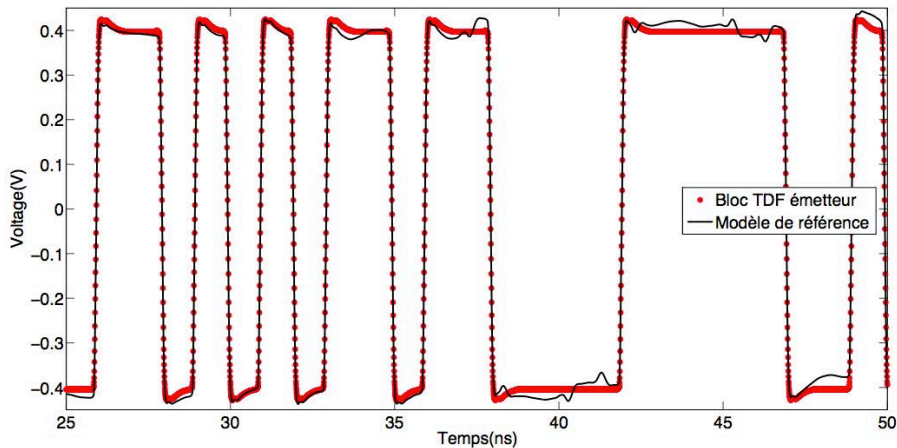


FIGURE 4.40 – Résultats de simulation du réseau de neurones pour l'émetteur + boîtier.

Nous avons également tracé le diagramme de l'œil des simulations pour analyser les performances de nos modèles. Le diagramme de l'œil est une méthode de représentation pour caractériser les performances d'une liaison numérique [Bre05]. Il correspond à la superposition sur un même intervalle de temps de l'ensemble des bits émis. La zone centrale où le signal ne passe jamais est appelée l'œil. De nombreux critères de performances peuvent être analysés par cette méthode. Par exemple, la période du signal, le temps de montée et le temps de descente sont observables, comme le montre la figure 4.44.a. Le diagramme de l'œil d'un système peut être, de plus, comparé à des masques normalisés

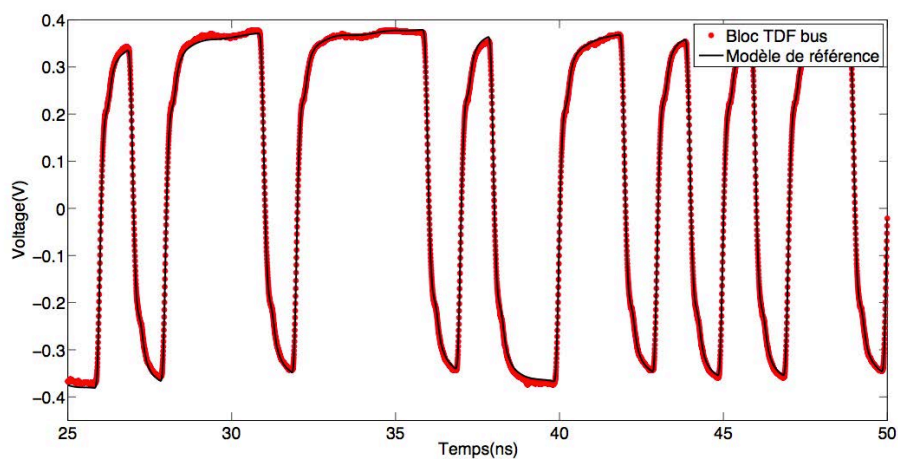


FIGURE 4.41 – Résultats de simulation du réseau de neurones pour le bus.

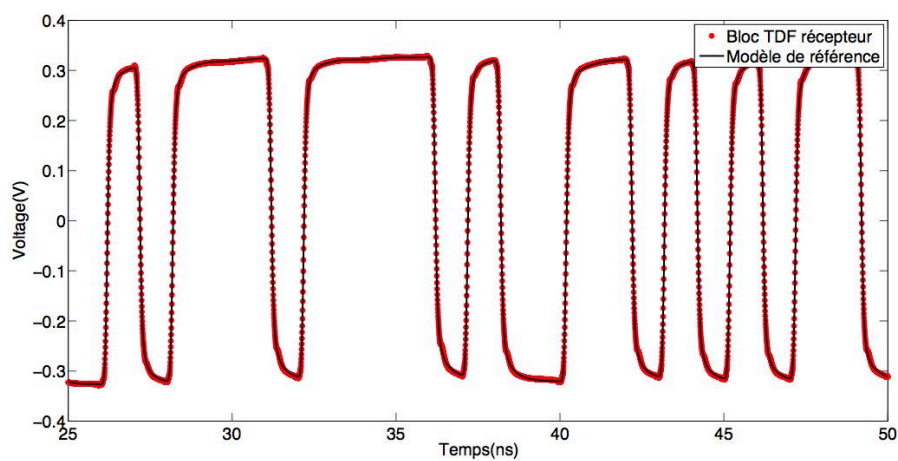


FIGURE 4.42 – Résultats de simulation du réseau de neurones pour le récepteur + boîtier.

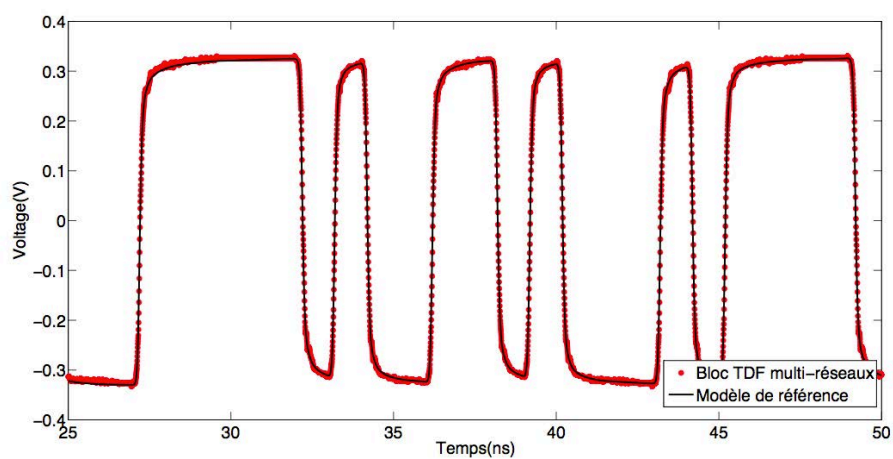


FIGURE 4.43 – Résultats de simulation du bloc multi-réseaux complet.

TABLE 4.6 – Précision des modèles du bloc multi-réseaux.

Réseau de neurones ou notre bloc	RAE (%)	Coefficient de corrélation
Réseau de neurones de l'émetteur	4	0.9992
Réseau de neurones du bus	1.69	0.9998
Réseau de neurones du récepteur	0.68	0.9999
Bloc multi-réseaux	1.48	0.9997

qui permettent de valider la qualité du signal par rapport à un protocole donné (figure 4.44.b). Un œil ouvert correspond à un signal comportant un minimum de déformations. Par contre, l'ISI ou le bruit conduisent à une fermeture de l'œil.

Pour tracer les diagrammes de l'œil de nos modèles, nous avons simulé la transmission d'une séquence de 250 bits, générée aléatoirement. Nous considérons tout d'abord notre application avec un niveau de pré-emphase à 0. La figure 4.45 montre le diagramme de l'œil du modèle de référence. Nous le comparons aux diagrammes du modèle mono-réseau (figure 4.46) et multi-réseaux (figure 4.47). Les trois diagrammes de l'œil sont obtenus en superposant les 250 bits de notre simulation. En comparant ces trois diagrammes de l'œil, nous constatons qu'ils ont des ouvertures similaires. Ces caractéristiques d'ouverture sont listées dans le tableau 4.7. Les écarts se situent en dessous des 5%. Des différences plus marquées apparaissent sur les transitions des signaux. Cette différence est la conséquence de l'apprentissage des réseaux de neurones. En effet, l'apprentissage du réseau de neurones est une procédure d'approximation, son objectif est de minimiser les différences entre les résultats simulés par le réseau et les sorties-désirées. Plutôt que d'apprendre toutes les valeurs des sorties-désirées dans tous les cas possibles, le réseau va avoir tendance à approximer chaque situation vers quelques valeurs typiques. En conséquence, les diagrammes de l'œil de nos modèles vont perdre des détails.

TABLE 4.7 – Ouvertures des diagrammes de l'œil (niveau 0)

Type de modèle	Ouverture verticale (V) et erreur relative (%)	Ouverture horizontale (ns) et erreur relative (%)
Modèle de référence	0.5924 et N/A	0.9825 et N/A
Bloc mono-réseau	0.6031 et 1.8	0.9878 et 0.5
Bloc multi-réseaux	0.6196 et 4.6	0.9918 et 0.9

Pour illustrer cela, nous avons enregistré les valeurs de tensions de tous les points situés à l'instant 1.11 ns de la période de transmission d'un bit, pour le modèle de référence (figure 4.48) et pour notre modèle mono-réseau (figure 4.49). Nous constatons que la distribution des valeurs de notre modèle est concentrée autour de quelques valeurs, alors que la distribution des valeurs du modèle de référence est plus dispersée.

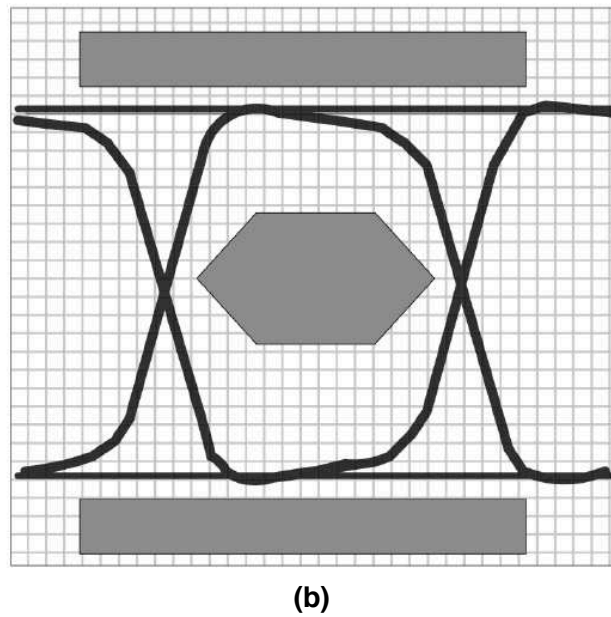
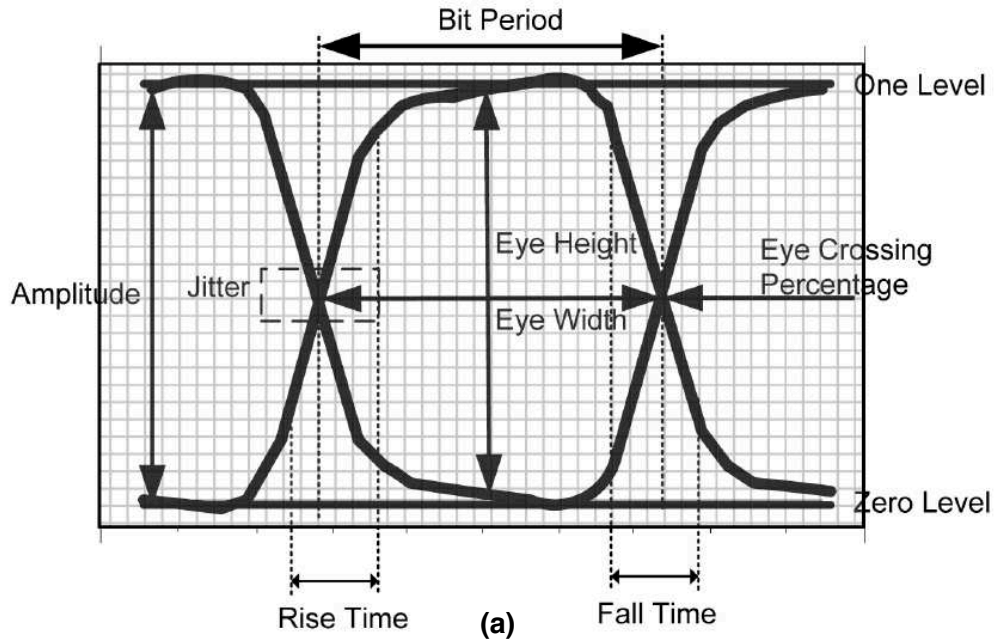


FIGURE 4.44 – Diagramme de l'oeil : (a) Mesure de performances (b) Masques normalisés (en gris) et mesure [eye12].

Nous avons également analysé le diagramme de l'œil avec un niveau 20 de pré-emphase du modèle de référence (figure 4.50) et de notre modèle mono-réseau paramétrable (figure 4.51). L'ouverture des diagrammes pour ces deux modèles est donnée dans le tableau 4.8. Nous observons les mêmes phénomènes que précédemment.

Enfin, les temps de simulation de nos modèles et du modèle de référence sont présentés dans le tableau 4.9. Notre méthodologie permet de diminuer considérablement le temps de simulation par rapport au modèle HSPICE à bas niveau. Les *speedups* selon les méthodes

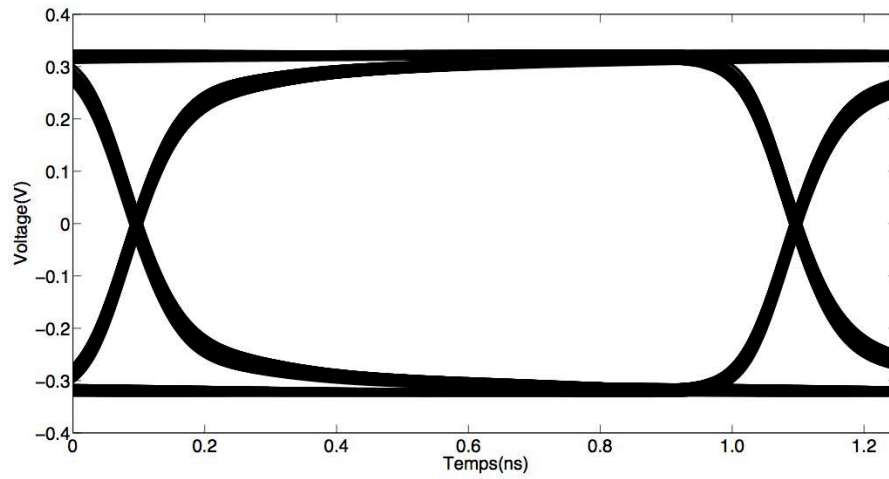


FIGURE 4.45 – Diagramme de l’oeil du modèle de référence (niveau 0).

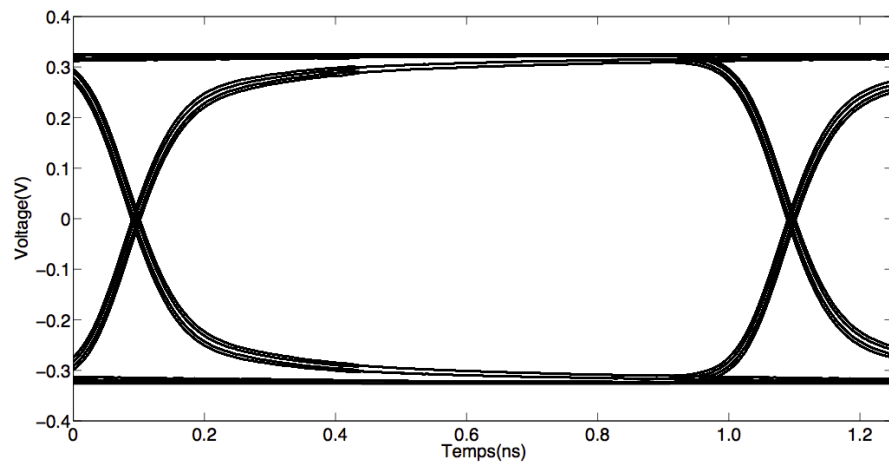


FIGURE 4.46 – Diagramme de l’oeil du bloc mono-réseau (niveau 0).

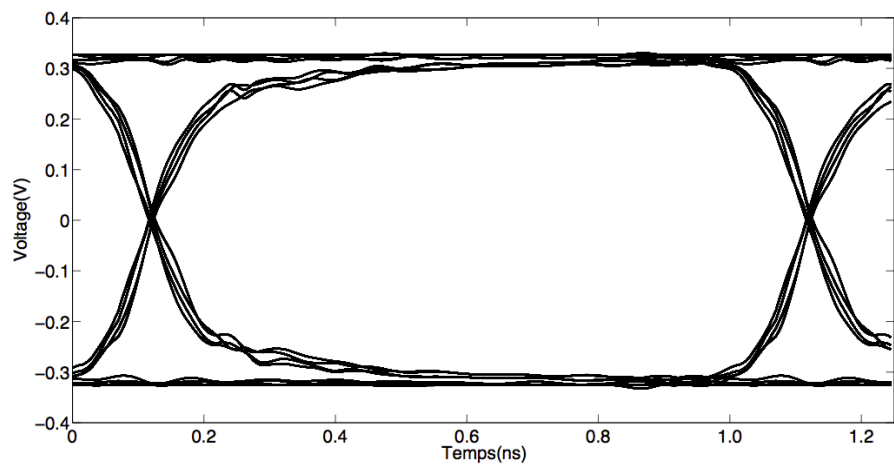


FIGURE 4.47 – Diagramme de l’oeil du bloc multi-réseaux (niveau 0).

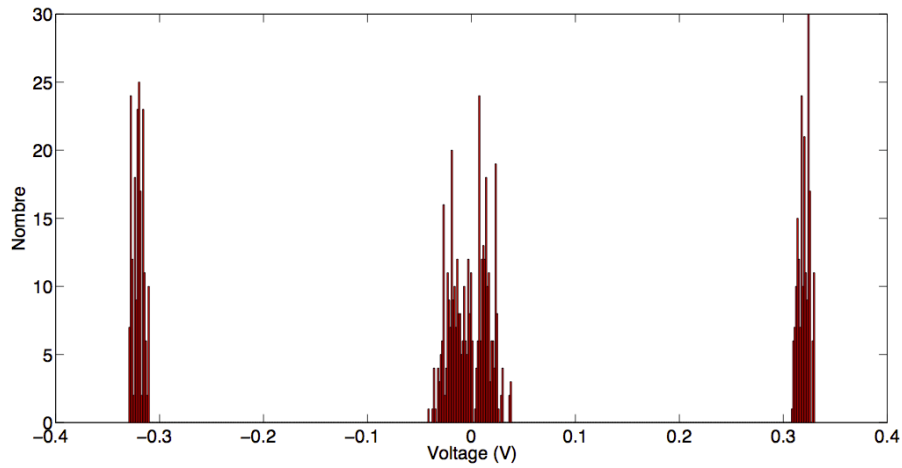


FIGURE 4.48 – Distribution des points à 1.11 ns du modèle de référence.

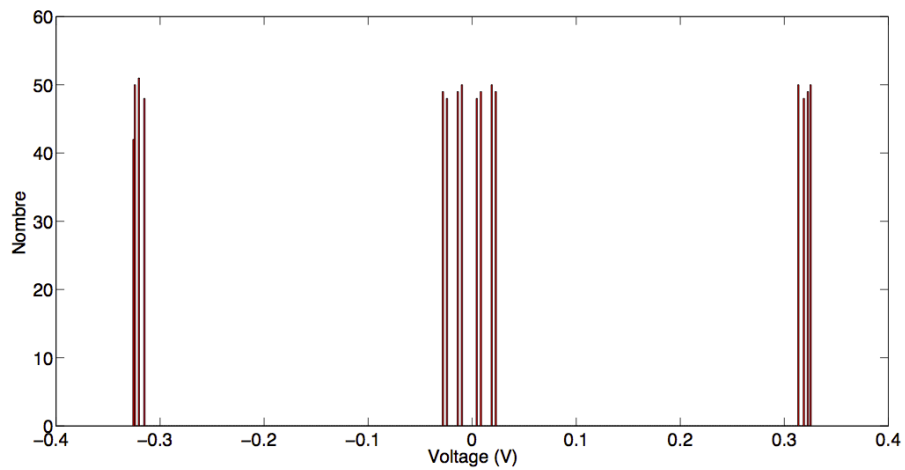


FIGURE 4.49 – Distribution des points à 1.11 ns de notre bloc mono-réseau.

TABLE 4.8 – Ouvertures des diagrammes de l'oeil (niveau 20)

Type de modèle	Ouverture verticale (V) et erreur relative (%)	Ouverture horizontale (ns) et erreur relative (%)
Modèle de référence	0.4836 et N/A	0.9729 et N/A
Bloc mono-réseau paramétrable	0.5004 et 3.5	0.9829 et 1

varient entre 1,910 et 8,775. Les meilleures performances sont atteintes par le modèle mono-réseau paramétrable, car ce réseau de neurones a une matrice W plus petite. La taille de cette matrice est donnée par l'équation 4.4 :

$$Taille_W = (\text{délais} + 1) \times \text{nombre d'entrées} \times \text{nombre de neurones}. \quad (4.4)$$

Le bloc mono-réseau paramétrable ayant un nombre de délais beaucoup plus faible que

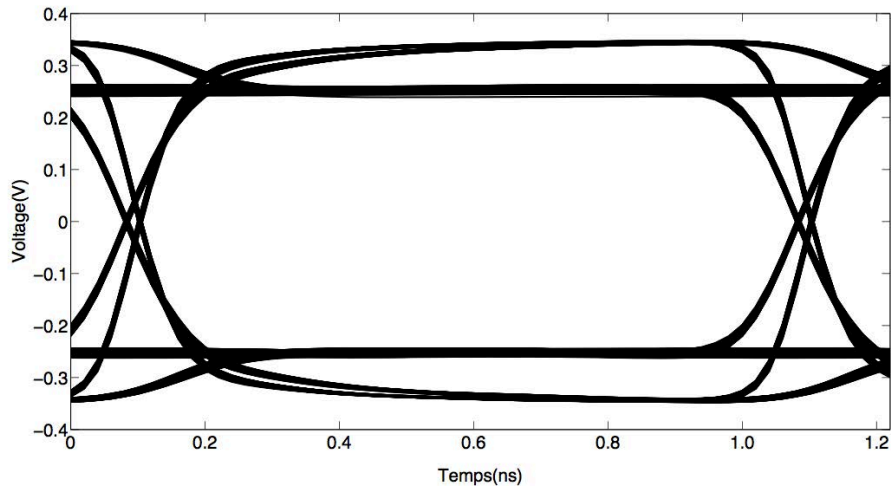


FIGURE 4.50 – Diagramme de l’oeil du modèle de référence (niveau 20).

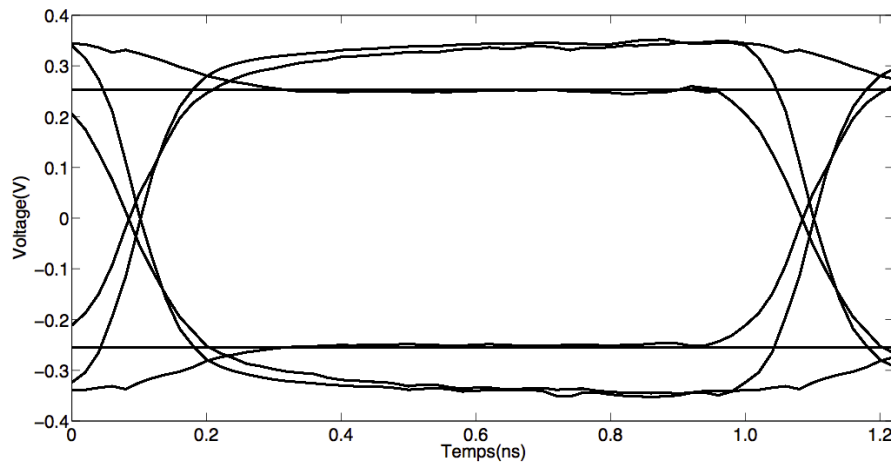


FIGURE 4.51 – Diagramme de l’oeil du bloc mono-réseau paramétrable (niveau 20).

les autres réseaux (cf. tableau 4.3), la matrice est donc plus petite et est traitée beaucoup plus rapidement lors de la simulation.

TABLE 4.9 – Temps de simulation des différents modèles

Type de modèle	Temps de simulation(<i>Intel Core i5-660 3.33GHz, RAM 2Gbits</i>)	Speedups
Modèle de référence	2,808.35 s	N/A
Bloc multi-réseau	1.47 s	×1,910
Bloc mono-réseau	1.10 s	×2,552
Bloc mono-réseau paramétrable	0.32 s	×8,775

4.3.5 Conclusion

Dans cette section, nous avons présenté notre deuxième cas d'étude, à savoir une application autour d'une liaison USB 3.0. Nous avons appliqué notre méthodologie de trois façons différentes, afin de répondre aux questions soulevées à la fin de la section 4.2 :

- La validation de notre bloc mono-réseau démontre la faisabilité de notre méthodologie pour des applications à haute fréquence, caractéristiques des protocoles de communication les plus récents.
- La validation de notre bloc mono-réseau paramétrable démontre que notre méthodologie permet de paramétrer un système plus efficacement. Grâce à notre modèle au niveau système, le concepteur peut déterminer la meilleure configuration pour ses composants dès le début de la conception. De plus, si l'on compare notre approche à des solutions existantes basées sur des modèles bas niveau, le temps nécessaire pour cette configuration est réduit de plusieurs ordres de grandeur.
- La validation de notre bloc multi-réseaux montre la possibilité d'appliquer un réseau de neurones par sous-bloc du système. Cette modularité de notre méthode permet de travailler avec des réseaux de neurones de complexité raisonnable, même pour une application complexe.

4.4 Conclusion

Dans ce chapitre, pour valider la méthodologie que nous avons développée, nous avons présenté et modélisé deux plateformes, un système communiquant à basse fréquence, via un bus I²C et un autre communiquant à haute fréquence, via une liaison USB 3.0.

En comparant nos modèles à haut niveau d'abstraction à d'autres approches de plus bas niveau, nous avons montré que notre méthodologie présentait quatre caractéristiques importantes :

- **Précision** : Nos modèles présentent une bonne précision, l'erreur par rapport aux modèles de référence n'a jamais été supérieure à quelques pourcents.
- **Vitesse** : La montrée en abstraction de nos modèles nous a permis d'atteindre des temps de simulation beaucoup plus courts que ceux des autres approches.
- **Flexibilité** : En comparaison avec la modélisation à bas niveau, notre méthode a besoin de moins de paramètres. Cependant, nous pouvons ajouter à nos modèles des paramètres d'entrée qui vont permettre au concepteur de configurer simplement et efficacement son système dès les premières phases de la conception.
- **Modularité** : Nous avons pu donner les caractéristiques d'intégrité du signal d'une application en appliquant un réseau de neurones par composant du système. Nous pouvons donc associer au modèle fonctionnel d'un composant son modèle non-fonctionnel, permettant ainsi de détecter d'éventuels problèmes d'intégrité du signal. On pourrait alors imaginer que nos modèles puissent enrichir une bibliothèque d'IPs permettant de prototyper virtuellement une application.

Chapitre 5

Conclusion et Perspectives

Sommaire

5.1 Conclusion	105
5.2 Perspectives	106

5.1 Conclusion

Nous avons, au cours de ces travaux de thèse, tenté de répondre à une problématique : comment modéliser, à haut niveau d'abstraction et dans un environnement unifié, les principaux phénomènes d'intégrité du signal, et comment les intégrer à un modèle fonctionnel d'un système hétérogène ?

Au début de nos travaux, nous avons recensé les méthodes et techniques existantes pour, d'une part, modéliser et simuler les systèmes hétérogènes et, d'autre part, pour analyser l'intégrité du signal. Nous avons constaté qu'il existait une multitude de langages de modélisation. Dans les langages existants, le langage SystemC-AMS nous semble a priori être une solution satisfaisante. Nous l'avons donc choisi pour modéliser nos systèmes hétérogènes. Parmi les différentes méthodes de modélisation de l'intégrité du signal, nous nous sommes intéressés plus particulièrement à la modélisation par réseau de neurones. Depuis une dizaine d'années, les réseaux de neurones sont utilisés pour la modélisation des systèmes électroniques et l'analyse de l'intégrité du signal, mais de tels modèles restent limités à un composant et ne sont pas appliqués à des systèmes plus complexes.

A l'aide de ces deux outils (SystemC-AMS et les réseaux de neurones), nous avons proposé une méthodologie de modélisation basée sur la combinaison de deux types de blocs complémentaires : premièrement, des blocs fonctionnels pour modéliser tous les comportements idéaux du système ; deuxièmement, des blocs non-fonctionnels qui servent à modéliser les comportements non-idéaux et donnent les performances d'intégrité du signal. Ces blocs représentent de façon analogique les valeurs logiques qui sont transférées sur le bus de communication. Les échanges d'informations entre blocs fonctionnels et non-fonctionnels sont assurés par des adaptateurs que nous avons également développés.

Nous avons appliqué cette méthodologie à deux cas d'étude : un système communiquant à bas débit via un bus I²C, et un système communiquant à haut débit via une liaison USB

3.0. Pour évaluer la justesse des résultats de nos modèles, deux modèles de références ont été mis en place. Ces deux applications ont permis, dans des contextes différents, de valider la faisabilité de notre méthode. Nous avons également montré l'avantage du moteur TDF dynamique pour la vitesse de simulation, et avons pu montrer la flexibilité et l'aspect modulaire de notre méthodologie. Enfin, grâce au haut niveau d'abstraction des modèles, notre méthode permet de réduire de façon remarquable le temps de simulation, tout en gardant une bonne précision.

Pour le concepteur, notre méthodologie offre ainsi trois avantages principaux :

- En comparaison avec un flot de conception traditionnel, notre méthode permet de modéliser et simuler conjointement les performances d'intégrité du signal et les fonctionnalités d'un système hétérogène avec un environnement unifié. Cette co-simulation répond dans une certaine mesure aux défis présentés dans la section 1.2. Notre méthode permet potentiellement de réduire le coût et aussi de raccourcir le temps de mise sur le marché.
- A l'aide de notre méthodologie, le concepteur peut configurer simplement et rapidement son système dès les premières étapes de la conception. De plus, nos modèles à haut niveau d'abstraction ont besoin de moins de paramètres d'entrée. Cela simplifie leur réalisation et permet en outre de protéger les informations sensibles du système.
- L'aspect modulaire de notre méthodologie offre la possibilité de bâtir ou d'enrichir une bibliothèque d'IPs à haut niveau d'abstraction. Chaque IP inclurait ainsi le bloc fonctionnel et le bloc non-fonctionnel. Une telle bibliothèque permettrait d'aider les concepteurs à analyser les performances d'intégrité du signal et à modéliser les fonctionnalités du système plus simplement et plus rapidement.

5.2 Perspectives

Nos travaux ouvrent un certain nombre de perspectives qui peuvent être regroupées autour de trois axes :

- Tout d'abord, nous pourrions approfondir la validation de notre méthode à l'aide de mesures. Dans ce manuscrit, nous avons vérifié notre méthode à l'aide de simulations de modèles de références. Cependant, la construction de nos modèles et leur validation à l'aide de mesures permettrait d'enrichir ces modèles en intégrant des phénomènes d'intégrité du signal non pris en compte par les modèles de références (par exemple, pour la plateforme I²C, les influences des champs électromagnétiques externes ne sont pas pris en compte).
- Ensuite, il serait intéressant d'exploiter la gamme d'utilisation de notre méthode. Ce point de perspective inclut deux domaines :
 1. Paramétrage : Nous avons, dans nos travaux, réussi à paramétrer nos modèles avec le niveau de pré-emphase d'un émetteur. Il serait intéressant de poursuivre sur cette voie en intégrant d'autres paramètres qui ont une influence sur l'intégrité du signal d'un système, comme la température, la longueur de la ligne de transmission, etc.
 2. Applications : Dans cette thèse, notre méthode a été appliquée à deux systèmes échangeant leurs données via des bus de communication. Ces deux systèmes ne contiennent que des sous-blocs numériques (processeur, RAM). Dans le futur, nous envisageons d'appliquer notre méthode à un système sur puce, ou à des

systèmes cyber-physiques, comme des capteurs sans fils incluant des composants analogiques ou des capteurs/actionneurs faisant l'interface avec d'autres domaines (comme l'optique ou la chimie).

- Enfin, il serait envisageable de développer une bibliothèque d'IPs telle que nous l'avons indiquée dans la section précédente. Pour bâtir cette bibliothèque, il faudrait cependant répondre d'abord à cette question : Est-il possible de développer des modèles totalement interopérables à partir de notre méthodologie ? Dans nos travaux, nous avons par exemple connecté notre modèle de câble USB 3.0 avec un modèle d'un certain type d'émetteur/récepteur (Altera, en l'occurrence). Si l'on décidait de remplacer cet émetteur par un autre type de composant et son modèle associé, pourrions-nous alors réutiliser en l'état le modèle du câble et obtenir une simulation correcte ? Ou bien est-ce que le réseau de neurones ne serait pas assez flexible pour cela ? Il serait alors nécessaire de retravailler sur l'architecture ou les techniques d'apprentissage de nos réseaux de neurones, ce qui ouvre de larges perspectives.

Publications

Conférences internationales avec actes et comité de lecture

- **DCIS'13**, R. Wang, J. Denoulet, S. Feruglio, F. Vallette and P. Garda, *High Level Modeling of Signal Integrity in a High Frequency Point-to-Point System with SystemC-AMS*, Conference on Design of Circuits and Integrated Systems (DCIS), Donostia-san Sebastian, Spain, Nov. 2013.
- **FDL'13**, R. Wang, J. Denoulet, S. Feruglio, F. Vallette and P. Garda, *High Level Modeling of Signal Integrity in bus communications with SystemC-AMS*, Forum on specification & Design Languages (FDL), Paris, Sept. 2013.
- **ICECS'12**, R. Wang, J. Denoulet, S. Feruglio, F. Vallette and P. Garda, *High Level Modeling of Signal Integrity in Field Bus Communication with SystemC-AMS*, IEEE International Conference on Electronics, Circuits and Systems (ICECS), Seville, Spain, Dec. 2012.

Conférences nationales avec actes et comité de lecture

- **GDR SoC-SiP'13**, R. Wang, J. Denoulet, S. Feruglio, F. Vallette and P. Garda, *High Level Modeling of Signal Integrity in High Frequency USB 3.0 Systems with SystemC-AMS*, Colloque du GDR SOC-SIP, Lyon, France, Jun. 2013.
- **GDR SoC-SiP'12**, R. Wang, J. Denoulet, S. Feruglio, F. Vallette and P. Garda, *Modeling Field Bus Communication with SystemC-AMS*, Colloque du GDR SOC-SIP, Paris, France, Jun. 2012.

Table des figures

1	Exemple de trois microprocesseurs d'Intel réalisés à des époques différentes [BC11].	1
1.1	Schéma de principe d'un système de contrôle automatique de l'état des pneus d'un véhicule.	6
1.2	Hétérogénéité architecturale : (a) Processeur + GPU et (b) Processeur + FPGA.[BDH ⁺ 10]	7
1.3	Système hétérogène matériel/logiciel.	8
1.4	Flot traditionnel de conception.	8
1.5	(a) Forme idéale d'un signal numérique et (b) Forme réelle de ce même signal.	10
1.6	Illustration de l'influence du délai de propagation.	11
1.7	Illustration du phénomène de désadaptation d'impédance.	12
1.8	Diaphonie entre deux lignes proches.	13
1.9	Phénomène de rebond du potentiel de masse.	14
2.1	(a) Flot de conception <i>HW/SW</i> traditionnel. (b) Flot de co-conception <i>HW/SW</i>	18
2.2	Niveaux d'abstraction pour la modélisation de circuits numériques et analogiques	20
2.3	Niveaux d'abstraction couverts par les langages VHDL et Verilog.	21
2.4	Organisation de SystemC.	22
2.5	Architecture de SystemC-AMS 1.0.	25
2.6	Comparatif des langages présentés.	26
2.7	Différents types de modèles de ligne de transmission.	28
2.8	Différents types de modèles de deux lignes de transmission couplées.	29
2.9	Structure d'un modèle IBIS d'une porte de sortie.	29
2.10	Exemple de modèle IMIC.	30
2.11	Exemple de modèle ICEM.	31

3.1	Schema de principe de notre méthodologie : (a) Blocs fonctionnels seuls et (b) Intégration des blocs non-fonctionnels avec les blocs fonctionnels.	37
3.2	Caractéristiques de MoCs de SystemC-AMS.	38
3.3	Forme des signaux dans les MoCs : (a) ELN, (b) LSF et (c) TDF	38
3.4	Deux descriptions différentes d'un amplificateur en SystemC-AMS TDF. . .	39
3.5	Structure d'un cluster TDF.	39
3.6	Exemples d'ordonnancements (a) cohérent et (b) incohérent.	40
3.7	Illustration d'une modulation de fréquence.	41
3.8	Module TDF conventionnel pour la modulation de fréquence.	42
3.9	Module TDF dynamique pour pour la modulation de fréquence.	42
3.10	Structure d'un modèle LSF.	43
3.11	Structure d'un modèle ELN.	43
3.12	Convertisseur de tension ELN vers TDF.	44
3.13	Schéma de principe d'un neurone biologique [wik].	45
3.14	a) Exemple de neurone artificiel, b) Symbole simplifié.	46
3.15	Exemples de fonctions d'activation (a) hard-limit, (b) identité, (c) log-sigmoïde, et (d) tan-sigmoïde.	46
3.16	Schéma de principe d'un réseau monocouche.	48
3.17	Schéma de principe d'un réseau multicouches non-bouclé.	48
3.18	Deux types de connexions : (a) entière et (b) partielle.	49
3.19	Exemple de réseau de neurones bouclé.	49
3.20	Etapes nécessaires à la construction d'un bloc non-fonctionnel.	53
3.21	Méthode d'acquisition des entrées/sorties.	54
3.22	Principe d'insertion de valeurs.	55
3.23	Fichier de stockage des couples d'entrées/sorties-désirées.	56
3.24	Architectures de réseaux de neurones proposés par le <i>Neural Network Toolbox</i> de MATLAB.	57
3.25	Forme générale d'un fichier ".nn".	58
3.26	Exemple général d'un bloc non-fonctionnel.	59
3.27	Principe de détection de seuil par les adaptateurs.	60
3.28	Principe d'insertion de valeurs intermédiaires par les adaptateurs.	61
4.1	Schéma électrique des E/S du bus I ² C (R _p : Résistance de rappel).	66
4.2	Trame de donnée I ² C.	66

4.3	Conditions START et STOP du protocole I ² C.	67
4.4	Procédure d'arbitrage sur le bus I ² C.	68
4.5	Exemple de système communiquant via un bus I ² C.	68
4.6	(a) Schéma original de SDA (b) Schéma implémenté en SystemC-AMS ELN [ADR ⁺ 08].	69
4.7	Résultats de simulation du modèle de référence.	70
4.8	Architecture du modèle de la plate-forme I ² C (ET : Porte "ET"; Insert : Insertion).	71
4.9	Bloc numérique du contrôleur de bus I ² C [Ala10].	72
4.10	Bloc fonctionnel pour l'application I ² C.	72
4.11	Circuit équivalent utilisé dans la première étape : (a) Interface avec le contrôleur de bus (b) Lignes de bus SDA et SCL (c) Résistances de charge.	73
4.12	Architecture du réseau NARX.	74
4.13	Pas de temps du modèle SystemC-AMS : (a) TDF conventionnel et (b) TDF dynamique.	75
4.14	Simulation du bloc fonctionnel lors d'une écriture.	76
4.15	Simulation du bloc fonctionnel (détail).	77
4.16	Simulation du bloc non-fonctionnel avec le moteur TDF conventionnel (dé- tail).	78
4.17	Simulation du bloc non-fonctionnel avec le moteur TDF dynamique (détail).	78
4.18	Superposition des résultats de nos différents modèles et du modèle réf- érence : (a) SCL (b) SDA.	79
4.19	Câble USB 3.0.	82
4.20	Architecture d'un système USB 3.0.	82
4.21	Architecture d'une connexion <i>SuperSpeed</i> USB 3.0.	83
4.22	Structure de l'en-tête.	84
4.23	Structure de la <i>payload</i>	84
4.24	(a) Système USB 3.0 à modéliser (b) Modèle de référence.	85
4.25	Illustration de l'ISI : (a) interférences et (b) pas d'interférences.	85
4.26	Filtre FIR de l'émetteur HSSI.	86
4.27	Construction du signal de pré-emphase avec le <i>1st Post-tap</i>	87
4.28	Signal issu de la fonction de pré-emphase pour différents niveaux de confi- guration du <i>1st Post-tap</i>	88
4.29	Architecture du bloc non-fonctionnel représentant la liaison USB 3.0 <i>SuperSpeed</i> (Insert : Insertion).	89
4.30	Signaux utilisés dans le bloc mono-réseau.	90

4.31 Signaux utilisés dans le bloc mono-réseau paramétrable.	90
4.32 Signaux utilisés dans le bloc multi-réseaux.	91
4.33 Architecture du réseau FTDNN.	91
4.34 Architecture du bloc non-fonctionnel mono-réseau.	92
4.35 Architecture du bloc mono-réseau paramétrable.	93
4.36 Relation entre niveau de pré-emphase et x	93
4.37 Architecture du bloc multi-réseaux.	94
4.38 Résultats de simulation du bloc mono-réseau.	95
4.39 Résultats de simulation du bloc mono-réseau paramétrable (niveau de pré-emphase : 20).	95
4.40 Résultats de simulation du réseau de neurones pour l'émetteur + boîtier. .	96
4.41 Résultats de simulation du réseau de neurones pour le bus.	97
4.42 Résultats de simulation du réseau de neurones pour le récepteur + boîtier. .	97
4.43 Résultats de simulation du bloc multi-réseaux complet.	97
4.44 Diagramme de l'oeil : (a) Mesure de performances (b) Masques normalisés (en gris) et mesure [eye12].	99
4.45 Diagramme de l'oeil du modèle de référence (niveau 0).	100
4.46 Diagramme de l'oeil du bloc mono-réseau (niveau 0).	100
4.47 Diagramme de l'oeil du bloc multi-réseaux (niveau 0).	100
4.48 Distribution des points à 1.11 ns du modèle de référence.	101
4.49 Distribution des points à 1.11 ns de notre bloc mono-réseau.	101
4.50 Diagramme de l'oeil du modèle de référence (niveau 20).	102
4.51 Diagramme de l'oeil du bloc mono-réseau paramétrable (niveau 20). . . .	102

Liste des tableaux

1	Exemples de protocoles de communication à grande vitesse.	2
2.1	Techniques de simulation de l'intégrité du signal	27
2.2	Outils pour l'intégrité du signal	32
3.1	Comparaison entre le module TDF conventionnel et le module TDF dynamique.	41
3.2	Bilan des MoCs de SystemC-AMS	45
3.3	Heuristiques de calcul du nombre de neurones (N) dans la couche cachée (N _i : nombre d'entrées, N _o : nombre de sorties.).	51
4.1	Paramètres de configuration de l'architecture du réseau NARX	74
4.2	Performances de nos modèles pour le système du bus I ² C	80
4.3	Paramètres de configuration des réseaux FTDNN pour les trois blocs non-fonctionnels.	92
4.4	Précision du bloc mono-réseau	95
4.5	Précision du bloc mono-réseau paramétrable	96
4.6	Précision des modèles du bloc multi-réseaux.	98
4.7	Ouvertures des diagrammes de l'oeil (niveau 0)	98
4.8	Ouvertures des diagrammes de l'oeil (niveau 20)	101
4.9	Temps de simulation des différents modèles	102

Bibliographie

- [Acc] Accellera. *SystemVerilog 3.1a Language Reference Manual Accellera's Extensions to Verilog*.
- [ADR⁺08] M. ALASSIR, J. DENOULET, O. ROMAIN, A. SUISSA et P. GARDA : Modeling field bus communications in mixed-signal embedded systems. *EURASIP Journal on Embedded Systems*, pages 1–11, 2008.
- [ADRG13] M. ALASSIR, J. DENOULET, O. ROMAIN et P. GARDA : Signal integrity-aware virtual prototyping of field bus-based embedded systems. *Components, Packaging and Manufacturing Technology, IEEE Transactions on*, 3(12):2081–2091, 2013.
- [AJK05] H. AL-JUNAID et T. KAZMIERSKI : Analogue and mixed-signal extension to SystemC. *Circuits, Devices and Systems, IEEE Proceedings*, pages 682–690, 2005.
- [AJK06] H. AL-JUNAID et T. KAZMIERSKI : HDL models of ferromagnetic core hysteresis using timeless discretisation of the magnetic slope. *In Design, Automation and Test in Europe (DATE). Proceedings*, volume 1, pages 1–2, 2006.
- [AJKW06] H. AL-JUNAID, T. KAZMIERSKI et L. WANG : SystemC-A modeling of an automotive seating vibration isolation system. *In Forum on Specification and Design Languages (FDL)*, pages 107–113, 2006.
- [Ala10] M. ALASSIR : *Modélisation au niveau système des entrées/sorties d'un système hétérogène*. Thèse de doctorat, UPMC - Sorbonne Universités, 2010.
- [Alt09] Altera Corporation. *Stratix IV GX HSSI Correlated HSPICE Model User Guide*, 2009.
- [Alt10] Altera Corporation. *Understanding the Pre-Emphasis and Linear Equalization Features in Stratix IV GX Devices*, november 2010.
- [Alt12] Altera Corporation. *Stratix IV Device Handbook Volume 2 : Transceivers*, 2012.
- [Alt14] Altera Corporation. *Stratix IV Device Handbook Volume 4 : Device Datasheet Data and Addendum*, March 2014.
- [Bai08] B. BAILEY : System level virtual prototyping becomes a reality with OVP donation from imperas. <http://www.ovpworld.org/system-level-virtual-prototyping-becomes-a-reality-by-brian-bailey>, 2008.
- [BC11] S. BORKAR et A. A. CHIEN : The future of microprocessors. *Commun. ACM*, 54(5):67–77, 2011.

- [BDH⁺10] A. R. BRODTKORB, C. DYKEN, T. R. HAGEN, J. M. HJELMERVIK et O. O. STORAASLI : State-of-the-art in heterogeneous computing. *Scientific Programming*, 18(1):1–33, 2010.
- [BEGTM11] M. BARNASCONI, K. EINWICH, C. GRIMM et A. Vachoux T. MAEHNE : *Advancing the SystemC Analog/Mixed-Signal (AMS) Extensions : Introducing Dynamic Timed Data Flow*. Open SystemC Initiative, September 2011.
- [BG11] R. BAHETI et H. GILL : Cyber-physical systems. *The Impact of Control Technology*, pages 161–166, 2011.
- [Bog03] E. BOGATIN : *Signal Integrity-Simplified*. Prentice Hall PTR, 2003.
- [Bre05] G. BREED : Analyzing signals using the eye diagram. *Journal of High Frequency Electronics*, novembre 2005.
- [Bro03] D. BROOKS : *Signal Integrity Issues and Printed Circuit Board Design*. Prentice Hall PTR, 2003.
- [BSA83] A.G. BARTO, R.S. SUTTON et C.W. ANDERSON : Neuronlike adaptive elements that can solve difficult learning control problems. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-13(5):834–846, 1983.
- [BSED07] H.G. BAKEER, O. SHAHEEN, H.M. EISSA et M. DESSOUKY : Analog, digital and mixed-signal design flows. *In Design and Test Workshop (IDT), 2nd International*, pages 247–252, 2007.
- [CB99] E. CHRISTEN et 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*, 46(10):1263–1272, 1999.
- [CBG90] S. CHEN, S.A. BILLINGS et P.M. GRANT : Non-linear system identification using neural networks. *Int. Journal of Contr.*, 51(6):1191–1214, 1990.
- [CDZ06] Y. CAO, R. DING et Q. ZHANG : State-space dynamic neural network technique for high-speed IC applications : modeling and stability analysis. *Microwave Theory and Techniques, IEEE Transactions on*, 54(6):2398–2409, 2006.
- [CEZ10] Y. CAO, I. ERDIN et Q. ZHANG : Transient behavioral modeling of nonlinear I/O drivers combining neural networks and equivalent circuits. *Microwave and Wireless Components Letters, IEEE*, 20(12):645–647, 2010.
- [CGTMS05] F.G. CANAVERO, S. GRIVET-TALOCIA, I.A. MAIO et I.S. STIEVANO : Analog and mixed-signal simulation of EMC at system level. *In Proc. of 16th IEEE International Zurich Symposium on Electromagnetic Compatibility, Zurich, CH*, 2005.
- [CJ10] D. CHEN et D. JIAO : An electromagnetics-based parallel transient simulator of linear complexity for the analysis of very large-scale integrated circuits and packages. *In Electromagnetic Compatibility (EMC), IEEE International Symposium on*, pages 396–400, 2010.
- [CSS11] F. CENNI, S. SCOTTI et E. SIMEU : SystemC-AMS behavioral modeling of a CMOS video sensor. *In VLSI and System-on-Chip (VLSI-SoC), IEEE/IFIP 19th International Conference on*, pages 380–385, 2011.
- [DBH09] H. DEMUTH, M. BEALE et M. HAGAN : *Neural Network Toolbox 6- User's Guide*. Mathworks, 2009.

- [DGG02] R. DOMER, A. GERSTLAUER et D. GAJSKI : *SpecC Language Reference Manual Version 2.0*. SpecC Technology Open Consortium, 2002.
- [dis10] VESA displayport standard version 1, revision 2, 1 2010.
- [Don63] W. M. DONALD : An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.
- [EHB93] R. ERNST, J. HENKEL et T. BENNER : Hardware-software cosynthesis for microcontrollers. *Design Test of Computers, IEEE*, 10(4):64–75, 1993.
- [EJL⁺03] J. EKER, J. JANNECK, E. A. LEE, J. LIU, X. LIU, J. LUDVING, S. SACHS et Y. XIONG : Taming heterogeneity - the Ptolemy approach. *Proceedings of the IEEE*, 91(1):127–144, 2003.
- [eld05] Eldo user’s manual, 2005.
- [eye12] Understanding data eye diagram methomethod for analyzing high speed digital signals. Rapport technique, Semiconductor Components Industries, 2012.
- [FYWZ00] Y. FANG, M. C. E. YAGOUB, F. WANG et Q. ZHANG : A new macromodeling approach for nonlinear microwave circuits based on recurrent neural networks. *Microwave Theory and Techniques, IEEE Transactions on*, 48(12):2335–2344, 2000.
- [GC90] Shepherd GM et Koch C : Introduction to synaptic circuits. *The Synaptic Organization of The Brain*, pages 3–31, 1990.
- [GCO⁺07] F. GAMBINI, M. CONTI, S. ORCIONI, F. RIPA et M. CALDARI : Physical modelling in SystemC-WMS and real time synthesis of electric guitar effects. *In Intelligent Solutions in Embedded Systems, Fifth Workshop on*, pages 87–100, 2007.
- [GD04] C. GARCIA et M. DELAKIS : Convolutional face finder : a neural architecture for fast and robust face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(11):1408–1423, 2004.
- [GDM93] R. K. GUPTA et G. DE MICHELI : Hardware-software cosynthesis for digital systems. *IEEE Des. Test*, 10(3):29–41, 1993.
- [GR00] G.G.E. GIELEN et R. A. RUTENBAR : Computer-aided design of analog and mixed-signal integrated circuits. *Proceedings of the IEEE*, 88(12), December 2000.
- [GT03] S. GRIVET-TALOCIA : Package macromodeling via time-domain vector fitting. *Microwave and Wireless Components Letters, IEEE*, 13(11):472–474, 2003.
- [GTHR⁺04] S. GRIVET-TALOCIA, H. HUANG, A.E. RUEHLI, F. CANAVERO et I.M. ELFADEL : Transient analysis of lossy transmission lines : an efficient approach based on the method of characteristics. *Advanced Packaging, IEEE Transactions on*, 27(1):45–56, 2004.
- [GTSC03] S. GRIVET-TALOCIA, I.S. STIEVANO et F.G. CANAVERO : Hybridization of FDTD and device behavioral-modeling techniques [interconnected digital I/O ports]. *Electromagnetic Compatibility, IEEE Transactions on*, 45(1):31–42, 2003.
- [Hay99] S. HAYKIN : *NEURAL NETWORKS : A Comprehensive Foundation*. Pearson Education, 1999.

- [HDB96] M.T. HAGAN, H.B. DEMUTH et M.H. BEALE : *Neural Network Design*. PWS Publishing Company, 1996.
- [hdm09] High definition multimedia interface specification version 1.4, 2009.
- [Heb49] D.O. HEBB : *The Organization of Behavior*. New York : Wiley, 1949.
- [hsp08] Hspice user guide : Simulation and analysis, 2008.
- [Hus89] D. HUSH : Classification with neural networks : a performance analysis. *In Systems Engineering, IEEE International Conference on*, pages 277–280, 1989.
- [HW01] F. HEUSCHEN et K. WALDSCHMIDT : Analog/digital co-design. 61:23–32, 2001.
- [ibi13] IBIS (I/O buffer information specification) version 6.0, 2013.
- [ILIF11] I.S.STIEVANO, L.RIGAZIO, I.A.MAIO et F.G.CANAVERO : Behavioral modeling of IC core power-delivery networks from measured data. *IEEE Trans. Compon., Packag., Manuf. Technol.*, 1(3):367–373, 2011.
- [imi01] EIAJ ED-5302 standard for I/O interface model for integrated circuits (IMIC), 2001.
- [Imp] Impulse accelerated technologies. *Impulse C User Guide*.
- [J.Z94] J.ZHENG : A 3D electromagnetic simulator for microwave and RF circuit boards. *In Electronic Components and Technology Conference. Proceedings.*, 44th, pages 550–556, 1994.
- [KS09] W. KAYANKIT et W. SUNTIAMORNTUT : Hardware/software co-design for line detection algorithm on FPGA. *In Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology,ECTI-CON. 6th International Conference on*, volume 01, pages 604–606, 2009.
- [KW97] I. KANELLOPOULOS et G. G. WILKINSON : Strategies and best practice for neural network image classification. *International Journal of Remote Sensing*, 18(4):711–725, 1997.
- [LBBH98] Y. LECUN, L. BOTTOU, Y. BENGIO et P. HAFFNER : Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [LDDS⁺08] C. LABUSSIÈRE-DORGAN, S.B. DHIA, E. SICARD, J. TAO, H.J. QUARESMA, C. LOCHOT et B. VRIGNON : Modeling the electromagnetic emission of a microcontroller using a single model. *IEEE Trans. Electromagn. Compat.*, 50(1):22–34, 2008.
- [Lev44] K. LEVENBERG : A method for the solution of certain problems in least squares. *Quart. Applied Math.*, 2:164–168, 1944.
- [LM93] D.I. LONG et S.S. MADHAT : Mixed-signal simulators : tools for mixed-signal design for test ? 1993.
- [LRP04] J. LEVANT, M. RAMDANI et R. PERDRIAU : ICEM modelling of microcontroller current activity. *Microelectronics Journal*, 35(6):501–507, 2004.
- [LWH90] K. J. LANG, A. H. WAIBEL et G. E. HINTON : A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3(1):23–43, 1990.

- [LZL⁺13] W. LI, D. ZHOU, M. LI, B. P. NGUYEN et X. ZENG : Near-field communication transceiver system modeling and analysis using SystemC/SystemC-AMS with the consideration of noise issues. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, PP(99), 2013.
- [Mat] MathWorks. *Simulink HDL Coder*.
- [MD09] A. E. MOUSSATI et C. DALLE : Developpement of a time-domain physical electromagnetic simulator for microwaves circuits. In *Microwave Symposium (MMS), 2009 Mediterranean*, 2009.
- [MG97] G. MICHELI et R. K. GUPTA : Hardware/software co-design. *Proceedings of the IEEE*, 85(3), 1997.
- [MRWSS07] P. MESSMER, V. RANJBAR, D. WADE-STEIN et P. SCHOESSOW : Advanced accelerator control and instrumentation modules based on FPGA. In *Particle Accelerator Conference (PAC). IEEE*, pages 506–508, 2007.
- [Nie87] R. H. NIELSEN : Kolmogorov’s mapping neural network existence theorem. In *Proceedings of the IEEE First International Conference on Neural Networks* (San Diego, CA), volume III, pages 11–13. Piscataway, NJ : IEEE, 1987.
- [NP73] L. W. NAGEL et D.O. PEDERSON : Spice(simulation program with integrated circuit emphasis), 1973.
- [NV13] P. NENZI et H. VOGT : Ngspice users manual version 25, January 2013.
- [nxp] Um1024 I²C-bus specification and user manual. http://www.nxp.com/documents/user_manual/UM10204.pdf.
- [OBC05] S. ORCIONI, G. BIAGETTI et M. CONTI : SystemC-WMS : A wave mixed signal simulator. In *8th Forum on specification Design Languages (FDL)*, 2005.
- [OCCH07] I O’CONNOR, B COURTOIS, K CHAKRABARTY et M HAMPTON : Heterogeneous systems on chip and systems in package. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2007.
- [Ope] Open SystemC Initiative. *SystemC 2.0.1 Language Reference Manual Revision 1.0*.
- [Ope98] Open Verilog International. *Verilog-AMS Language Reference Manual : Analog & Mixed-Signal Extensions to Verilog HDL*, 1998.
- [osc10] Open SystemC Initiative (OSCI) standard SystemC-AMS extensions language reference manual, 2010.
- [Pau06] C.R. PAUL : *Introduction to Electromagnetic Compatibility*. John Wiley & Sons, 2006.
- [Phi99] Philips Semiconductors. *Single-chip 8-bit microcontroller*, 1999.
- [Phi00] Philips Semiconductors. *The I²C-Bus Specification-version 2.1*, 2000.
- [RCG04] O. ROMAIN, T. CUENIN et P. GARDA : Design and verification of mixed-signal I/O IPs : an I²C bus controller. In *Industrial Electronics, 2004 IEEE International Symposium on*, volume 1, pages 77–81, 2004.
- [Rip93] B. D. RIPLEY : Statistical aspects of neural networks. *Networks and Chaos Statistical and Probabilistic Aspects*, 50:40–123, 1993.
- [Sca85] L.E. SCALES : *Introduction to Non-linear Optimization*. New York : Springer-Verlag, 1985.

- [Sch10] P. R. SCHAUMONT : *A Practical Introduction to Hardware/Software Code-sign*. Springer, 2010.
- [SH00] J.A. McCall S.H. HALL, G.W. Hall : *High-Speed Digital System Design-A Handbook of Interconnect Theory and Design Practices*. John Wiley & Sons, 2000.
- [Smi96] D. J. SMITH : VHDL & Verilog compared & contrasted - plus modeled example written in VHDL, Verilog and C. *In Design Automation Conference*, 1996.
- [soc] *A modelisation & simulation platform for system on chip*. <http://www.soclib.fr/trac/dev>.
- [SSCM08] I.S. STIEVANO, C. SIVIERO, F.G. CANAVERO et I.A. MAIO : Behavioral modeling of digital devices via composite local linear state-space relations. *Instrumentation and Measurement, IEEE Transactions on*, 57(8):1757–1765, 2008.
- [Swa01] S. SWAN : An introduction to system level modeling in SystemC 2.0. Rapport technique, Cadence Design Systems, 2001.
- [sys10] SystemC-AMS extensions user's guide. <http://www.accellera.org/downloads/standards/systemc/ams>, 2010.
- [sys12] Draft standard SystemC-AMS extensions 2.0 language reference manual, 2012.
- [Tei12] J. TEICH : Hardware/software codesign : The past, the present, and predicting the future. *Proceedings of the IEEE*, 100(Special Centennial Issue):1411–1430, 2012.
- [TT92] R. R. TRIPPI et E. TURBAN, éditeurs. *Neural Networks in Finance and Investing : Using Artificial Intelligence to Improve Real World Performance*. McGraw-Hill, Inc., New York, NY, USA, 1992.
- [usb11] Universal serial bus 3.0 specification. http://www.gaw.ru/pdf/interface/usb/USB30_english.pdf, May 2011.
- [usb13] USB 3.0 - A global strategic business report. Rapport technique, Global Industry Analysts, Inc., 2013.
- [UTE] UTE 47A EMC. *Integrated Circuit Electromagnetic Model (ICEM)*.
- [Vas05] G. VASILESCU : *Electronic noise and interfering signals : principles and applications*. Springer Berlin Heidelberg, 2005.
- [ver] IEEE standard for Verilog hardware description language.
- [VGE03] A. VACHOUX, C. GRIMM et K. EINWICH : SystemC-AMS requirements, design objectives and rationale. *In Design, Automation and Test in Europe Conference and Exhibition.*, pages 388–393, 2003.
- [VGE05] A. VACHOUX, C. GRIMM et K. EINWICH : Extending SystemC to support mixed discrete-continuous system modeling and simulation. *In Circuits and Systems (ISCAS). IEEE International Symposium on*, pages 5166–5169 Vol. 5, 2005.
- [VGKM06] A. VACHOUX, C. GRIMM, R. KAKEROW et C. MEISE : Embedded mixed-signal systems : New challenges for modeling and simulation. *In IEEE International Symposium n Circuits and Systems, ISCAS*, 2006.
- [vhd] IEEE standard VHDL language reference manual.

- [VNZ97] A. VELUSWAMI, M.S. NAKHLA et Q. ZHANG : The application of neural networks to EM-based simulation and optimization of interconnects in high-speed vlsi circuits. *Microwave Theory and Techniques, IEEE Transactions on*, 45(5):712–723, 1997.
- [VPB⁺08] M. VASILEVSKI, F. PECHEUX, N. BEILLEAU, H. ABOUSHADY et K. EINWICH : Modeling and refining heterogeneous systems with SystemC-AMS : Application to WSN. In *Design, Automation and Test in Europe (DATE)*, pages 134–139, 2008.
- [Wad91] B. C. WADELL : *Transmission line design handbook*. Artech House, 1991.
- [WDZ98] F. WANG, V. K. DEVABHAKTUNI et Q. ZHANG : A hierarchical neural network approach to the development of a library of neural models for microwave design. *Microwave Theory and Techniques, IEEE Transactions on*, 46(12):2391–2403, 1998.
- [WH08] Z. WANG et O. HAMMAMI : C-based hardware-accelerator coprocessing for SOC an quantitative area-performance evaluation. In *Int. Conf. on Electron., Circuits and Syst., ICECS*, 2008.
- [wik] neuron hand-tuned. [http://commons.wikimedia.org/wiki/File:
%3ANeuron_Hand-tuned.svg](http://commons.wikimedia.org/wiki/File:%3ANeuron_Hand-tuned.svg).
- [WS92] D. WHITE et D. SOFGE : *Handbook of Intelligent Control*. New York : Van Nostrand Reinhold, 1992.
- [You01] B. YOUNG : *Digital signal integrity : modeling and simulation with interconnects and packages*. Prentice Hall PTR, 2001.
- [ZG00] Q. ZHANG et K.C. GUPTA : *Neural Networks for RF and Microwave Design*. Norwood, MA : Artech House, 2000.

