# Systèmes neuromorphiques temps réel : contribution a l'intégration de réseaux de neurones biologiquement réalistes avec fonctions de plasticité

Bilel Belhadj

N° d'ordre : 4051

# THÈSE

PRESENTEE A

# L'UNIVERSITÉ BORDEAUX I

ECOLE DOCTORALE DES SCIENCES PHYSIQUES ET DE L'INGENIEUR

par **Bilel BELHADJ-MOHAMED**

POUR OBTENIR LE GRADE DE

# DOCTEUR

SPÉCIALITÉ : ELECTRONIQUE

---

## Systèmes neuromorphiques temps réel : contribution a l'intégration de réseaux de neurones biologiquement réalistes avec fonctions de plasticité

---

Soutenance le : 22 Juillet 2010

Après avis des rapporteurs :

| | | | |
|---|---|---|---|
| M. | Bertrand GRANADO | Professeur | ENSEA |
| M. | Laurent FESQUET | Maître de conférences, HDR | INP Grenoble |

Devant la commission d'examen composée de :

| | | | | |
|---|---|---|---|---|
| M. | Yannick BORNAT | Maître de conférences | ENSEIRB-IPB | Invité |
| M. | Dominique DALLET | Professeur | ENSEIRB-IPB | Président |
| M. | Laurent FESQUET | Maître de conférences, HDR | INP Grenoble | Rapporteur |
| M. | Bertrand GRANADO | Professeur | ENSEA | Rapporteur |
| Mme | Sihem GUEMARA | Professeur | SUP'COM | Examinatrice |
| Mme | Sylvie RENAUD | Professeur | ENSEIRB-IPB | Directrice |
| M. | Jean TOMAS | Maître de conférences | Université Bordeaux 1 | Co-directeur |

– 2010 –

# *Remerciements*

Je remercie, au premier lieu, mon *Dieu* qui m'a offert et préservé une bonne santé et qui m'a entouré de sa bienveillance et sa grâce. Je le remercie également de m'avoir confié à des gens respectueux, responsables et scientifiques pendant ces années de thèse.

Je remercie le professeur *Pascal Fouillat* pour m'avoir accueilli au sein du laboratoire IMS pendant la réalisation de ces travaux.

Merci également au professeur *Sylvie Renaud*, pour m'avoir accordé cette opportunité d'effectuer ces travaux de thèse au sein de son groupe de recherche et de m'avoir introduit à une large communauté scientifique nationale et internationale du domaine des neurosciences. Un grand merci aussi pour la bonne ambiance qu'elle veille à diffuser entre les membres de l'équipe : les innombrables gâteaux, le chocolat ainsi que l'humour qui les accompagne en sont des bons témoins.

Merci à *Jean Tomas* de m'avoir encadré de près durant ces années. Sa sagesse et son sens des responsabilités étaient une source de sérénité, de discussions fructueuses et de l'humour pour toute l'équipe. Mais surtout l'écoute, la compréhension et la confiance en toutes situations qui auront influencé ma formation tout ce temps.

Merci à *Yannick Bornat* pour son soutien moral, sa précieuse aide technique, sa disponibilité, sa précipitation à fournir de l'aide dans les moments difficiles et, plus que tout, sa patience lors des discussions imprévisibles qui, malgré nous, durent quelques heures contre les quelques minutes pré-annoncées. Merci beaucoup.

Je remercie les professeurs *Bertrand Granado* et *Laurent Fesquet* pour le temps qu'ils m'ont consacré en tant que rapporteurs de thèse.

Merci à *Sihem Guemara* et *Dominique Dallet* pour l'intérêt qu'ils ont porté à mon travail en acceptant de participer au jury de thèse.

Un merci à *Sylvain Saïghi* et *Noëlle Lewis*, travailler avec vous est un vrai plaisir. J'ai appris beaucoup de choses de votre bon sens et savoir faire.

*Olivia Malot*, l'ingénieur qui m'a accompagné dans les réalisations de la partie technique de ces travaux. Notre collaboration fut très fructueuse et enrichissante. Je te souhaite une bonne continuation dans ta vie professionnelle.

Une pensée aussi pour tous ceux qui ont contribué à ce travail, notamment les stagiaires que j'ai encadré comme *Lionel*, *Vivien*, *Frédéric*, *Dominique*, *Jonathan*, *Muammer* et *Youssef* mais aussi les stagiaires que je n'ai pas encadré comme *Arnaud*, *Jean Baptiste*, *Romain* et *Tamara*.

J'aimerais remercier l'ensemble du *personnel du laboratoire IMS* pour leur accueil, leurs services et leurs sourires quotidiens.

A tous mes collègues : *Guilherme*, *Youssef*, *Adam*, *Laure*, *Filippo*, *Adel*, *Adeline* et *Gilles*, merci de votre soutien et les bons moments qu'on a passé ensemble.

Je remercie aussi les membres de *l'équipe Bio-EM* pour l'apport scientifique, les chaleureuses réunions et la bonne coopération. Je vous souhaite beaucoup de réussite.

Voilà, le grand moment est venu, il s'agit de remercier *mes parents* ; sans leur soutien, je ne serais pas capable de surmonter les difficultés de cette vie.

Finalement, un merci pour tous ceux qui ont animé ma vie en dehors du cadre du travail : *Soumaya*, *Saif*, *Aladin (wildi)*, *Najmeddine (Ami)*, *Najeh (Ami)*, *Sahbi*, *Charlotte*, *Gilles*, *Moutassar (Mounta)*, *Hassène*, *Camille*, *Joëlle*, *Hassène*, *Moez*, *Mouhamed-Hicham (Momo)*, *Ali*, *Younes*, *François*, *Atidel*, *Mohamed (el masri)*, *Mahmoud*, *Si-Mohamed* et plein d'autres dont les noms ne sont pas présents dans mon esprit à l'instant mais à qui, certainement, je pense.

# *Table des matières*

# Introduction générale

Plusieurs disciplines sont mises en œuvre pour l'étude du fonctionnement, de la structure et du développement du système nerveux, telles que la biologie, l'informatique, l'électronique,... Les moyens techniques de ces différentes disciplines coopèrent pour un même objectif : découvrir les dynamiques neuronales. Dès lors, les spécialités disparaissent et une science nouvelle apparaît, qui se définit, non plus par les moyens techniques, mais par l'objet d'étude : le système nerveux. Baptisée neurosciences, cette nouvelle science réunit des efforts interdisciplinaires pour mieux comprendre le fonctionnement neuronal. Des descriptions partielles sur la structure et le fonctionnement du tissu nerveux existent. Un état de connaissances est donc disponible pour la communauté scientifique sous forme de bases de données concernant l'anatomie et les mécanismes du système nerveux. Cependant, toutes ces connaissances ne sont pas suffisantes pour permettre d'expliquer les paradigmes de calcul remarquablement employés par le cerveau.

La *modélisation* représente une approche importante dans le spectre des efforts d'étude des dynamiques neuronales. En effet, modéliser un élément pour qu'il se comporte d'une manière semblable à celui que l'on veut étudier est une solution pour des nombreux cas d'expériences neurobiologiques irréalisables, trop coûteuses ou soumises à des considérations éthiques. Une large communauté de chercheurs développe des modèles pour différentes parties du système nerveux dans le but, entre autres, d'en comprendre les mécanismes sous-jacents. Le niveau de détails des modèles change selon le mécanisme étudié.

La majorité des modélisateurs utilise des ordinateurs pour simuler numériquement des mécanismes neuronaux ainsi modélisés. Du fait des avantages du développement logiciel et l'évolution croissante de la puissance de calcul des processeurs, l'importance attribuée à ce domaine croit régulièrement. Cependant, les processeurs basés sur le paradigme de Turing (Turing, 1937), qui fonctionnent à la base des transitions séquentielles transitant entre des états internes discrets, sont en désaccord avec le calcul massivement parallèle et asynchrone inhérent aux systèmes neuronaux. Cette branche de modélisation présente donc des limites en termes de performances et de compatibilité avec les données biologiques.

Une approche alternative tire ses origines des analogies existantes entre la physique des semi-conducteurs VLSI[1] et la biophysique, souvent référencée par le terme *ingénierie neuromorphique* (Mead, 1990 ; Gauwenberghs, 1999). Des systèmes, dits *neuromorphiques*, proposent de reproduire la structure et le fonctionnement des systèmes neuronaux biologiques pour transférer leur capacité de calcul sur silicium. Ils offrent la possibilité de distribuer le calcul sur des unités analogiques élémentaires, parallèles et densément connectées. Depuis les travaux novateurs de Carver Mead (Mead, 1988), l'ingénierie neuromorphique ne cessent d'émerger des implémentations remarquables de systèmes biologiques, par exemple, la rétine sur Silicium (Delbrück, 2004), l'autoréglage des unités motrices pour les robots (Lewis, 2000), l'auto-organisation des patterns des classificateurs (Häfliger, 2007) et bien d'autres.

Bien que plusieurs applications soient prometteuses, l'ingénierie neuromorphique reste loin d'atteindre les capacités du calcul neuronal des systèmes biologiques. La plupart des efforts de conception sont plutôt orientés vers une application limitée. Jusqu'à nos jours, la

[1] Very-Large-Scale Integration, i.e. des circuits contenant des milliers de millions de transistors sur une même puce.

technologie et les techniques d'implémentation ne permettent pas encore de concevoir un dispositif neuromorphique qui soit suffisamment puissant et flexible pour servir comme outil de simulation pour les neurosciences contournant des outils logiciels.

Dans la dernière décennie, des efforts de conception et de modélisation ont permis de réaliser des architectures matérielles hautement configurables permettant de reproduire le fonctionnement et la structure d'une grande gamme de réseaux neuronaux. L'efficacité de ces architectures se mesure par la fidélité au biologique, la taille des réseaux, la flexibilité de la configuration et la précision du calcul.

Le brassage entre les connaissances en biologie, en électronique et en informatique complique la conception et la réalisation de tels systèmes. Des initiatives de rassemblement des chercheurs de domaines différents s'avèrent nécessaires. Le projet de recherche européen FACETS, introduit dans la section 1 du Chapitre 2, a été mis en place dans cette optique. Il réunit des chercheurs de plusieurs disciplines dans le but de réaliser des dispositifs innovants et flexibles qui serviront comme outils de calcul autonomes bio-inspirés.

**La contribution de cette thèse**

Le travail présenté dans cette thèse s'est fait dans la continuité d'autres travaux de l'équipe. Les circuits et les systèmes développés sont soumis à des contraintes temporelles et fonctionnelles pour s'approcher des comportements biologiques. L'application de ces contraintes nous permettra d'interfacer le système avec le monde extérieur, en particulier, avec des cellules nerveuses vivantes. Cette application fut le premier objectif de l'équipe. Raison pour laquelle, on s'intéresse particulièrement à satisfaire les contraintes relatives à l'évolution en temps réel biologique d'une façon stricte, ainsi qu'à les modèles biophysiquement réalistes des neurones et des fonctions d'apprentissage ; ceci au contraire de simulateurs de très larges réseaux neuronaux, s'intéressant aux fonctions cognitives et fonctionnement sans interaction avec le vivant.

Notre équipe de recherche développe des prototypes d'outils de modélisation neuromorphiques. L'approche suivie consiste à réaliser des systèmes purement matériels qui reproduisent le fonctionnement de plusieurs types de réseaux de neurones impulsionnels en temps réel biologique ; le niveau de réalisme prime sur la taille des réseaux, les paramètres de configuration des cellules possèdent un sens de point du vue biophysique et les modèles utilisées sont dits biologiquement réalistes.

Les cœurs de calculs des neurones sont réalisés en électronique analogique pour profiter des analogies avec le vivant, en particulier l'évolution continue du potentiel de membrane en temps réel. L'électronique numérique intervient dans la configuration, la réalisation de la plasticité du réseau, la communication des unités de calcul ainsi que l'interfaçage avec le logiciel utilisateur.

Les travaux de cette thèse portent sur la conception et la réalisation de la partie numérique du système. Il s'agit d'une contribution aux objectifs suivants :

- Conception d'architectures connexionnistes qui permettent de représenter la densité de la connectivité des réseaux de neurones biologiques,
- Intégration matérielle flexible et précise des modèles de plasticité synaptique,
- Communication temps-réel entre les neurones d'un réseau artificiel,
- Intégration, interfaçage et gestion d'un grand nombre de neurones sur circuits intégrés analogiques dans un même système matériel.

Ce manuscrit propose un ensemble de méthodes pour intégrer des neurones artificiels sur des plateformes de simulation matérielles neuromorphiques fonctionnant en temps réel biologique. A cette fin, des circuits intégrés analogiques qui calculent l'activité électrique de neurones, développées par l'équipe, ont été intégrés sur des cartes électroniques pour constituer un réseau. Les liaisons inter-neurones sont régies par des lois de plasticité synaptique calculées numériquement sur un FPGA[1].

**Réalisations techniques**

La première partie de la thèse présente la réalisation de la partie numérique d'un système neuromorphique constitué d'une carte électronique contenant 5 circuits analogiques spécifiques (ou ASIC[2]) et un FPGA. Un ASIC comprend 5 neurones qui sont capables de fonctionner en temps-réel biologique. Le FPGA assure la communication entre les neurones et calcule la plasticité des connexions synaptiques. Le travail réalisé est résumé dans les points suivants :

- Conception et réalisation d'une architecture connexionniste pour la simulation des réseaux de neurones impulsionnels,
- Implémentation sur FPGA d'un modèle réaliste de la plasticité synaptique,
- Mise en œuvre d'une plateforme de simulation de 25 neurones répartis sur 5 ASICs.

Les travaux présentés dans la deuxième partie visent à élargir la taille des réseaux en termes de nombre de neurones. Une architecture multicarte a été mise au point pour multiplier les supports physiques des ASICs et des FPGAs. Une carte contient 4 ASICs (20 neurones) et un FPGA. Au maximum, une vingtaine de cartes peuvent coopérer en temps réel pour réaliser une simulation. Les points suivants résument la contribution de cette thèse dans la réalisation de ce système :

- Conception et réalisation d'une architecture connexionniste distribuée pour la simulation des réseaux de neurones distribués sur plusieurs cartes électroniques,
- Implémentation d'une version distribuée du calcul de la plasticité synaptique sur des FPGAs localisés sur des cartes différentes,
- Définition, conception et implémentation d'un protocole de communication de type anneau à jeton qui garantit l'aspect temps-réel des interactions entre les neurones des cartes différentes,
- Mise en œuvre de la plateforme de simulation contenant 400 neurones.

Les chapitres dédiés à ces deux systèmes développeront ces points avec plus de détails.

Les travaux de cette thèse ont contribué à la réalisation d'un prototype. Le but ici n'est pas d'atteindre une maturité technologique dans la réalisation des systèmes, mais plutôt de prouver la faisabilité et évaluer les performances des techniques et des méthodes proposées pour l'intégration des réseaux de neurones en matériel. Des méthodes et des solutions pratiques ont été développées pour surmonter les problèmes conceptuels et techniques liés à la réalisation des systèmes neuromorphiques. Les idées ainsi développées seront réutilisées pour le développement des systèmes futurs.

[1] Le FPGA (Field Programmable Gate Array) est un circuit numérique configurable.
[2] Application Specific Integrated Circuit

**Structure de la thèse**

Le manuscrit est structuré comme suit : après cette introduction générale, le Chapitre 1 présente les domaines des neurosciences et l'ingénierie neuromorphique. Une exposition des notions de bases de la neurobiologie ainsi que les principaux modèles qui les représentent font l'objet des sections 2 et 3. Un état de l'art des systèmes matériels utilisés comme des outils de modélisation neuromorphiques termine le chapitre. Le Chapitre 2 résumera l'historique et l'évolution de la conception des circuits et systèmes neuromorphiques au sein de notre groupe de recherche. On mettra l'accent sur l'évolution d'un système nommé PAX qui, dans ses versions 2 et 3, a été sujet de la contribution de cette thèse. Les Chapitres 3 et 4 détailleront la réalisation du système PAX respectivement dans les versions 2 et 3. Chaque chapitre commence par une définition du cahier des charges de la version du système à réaliser. Ensuite, les principales techniques et méthodes de réalisation sont exposées ainsi que l'étude de leurs performances de fonctionnement. Enfin, l'étape de mise en réseau des différents composants du système est illustrée à la fin de chaque chapitre. A la fin du manuscrit, une conclusion générale du travail réalisé est fournie avec une perspective sur le travail futur.

**Conventions utilisées**

Dans cette partie, on précisera les sens de quelques termes qui, sans explications, peuvent prêter à confusion.

Le sujet de cette thèse se focalise sur les techniques d'implémentation des modèles neuronaux à l'aide des circuits et systèmes neuromorphiques. Ces modèles seront par la suite l'objet d'une simulation. Comme le terme simulation fait généralement référence à une résolution d'un ensemble d'équations mathématiques à l'aide d'un programme informatique et donc de caractère numérique, on utilisera le terme *simulation matérielle* pour faire référence à une simulation réalisée avec une plateforme matérielle (émulation matérielle par le biais des systèmes neuromorphiques).

Dans la même logique, on appellera *modélisation neuromorphique* la manière selon laquelle un modèle neuronal est représenté sur un circuit ou un système neuromorphique.

On utilisera le terme *temps réel biologique* tout au long de ce manuscrit. Dans le cas général, un système fonctionnant en temps réel est soumis à une dépendance entre son déroulement et le déroulement d'un phénomène physique ayant une cadence propre. Dans notre cas, ce phénomène n'est que l'évolution de l'information neuronale : la dynamique de la formation des potentiels d'action, les mécanismes synaptiques, la propagation de l'information dans les axones et les dendrites, etc. Il nous est donc nécessaire de garantir que les temps nécessaire à la propagation de l'information et aux calculs de mises à jour des paramètres du système n'affectent pas son activité.

La dernière convention concerne la définition des systèmes neuromorphiques. L'évolution croissante de l'ingénierie neuromorphique a fait évoluer les méthodes et les techniques de réalisation. Au départ, les systèmes neuromorphiques faisaient référence aux circuits analogiques qui reproduisent une fonction et/ou une structure neuronale. Ensuite, ils ont évolué vers des systèmes mixtes analogique-numérique pour couvrir des besoins en flexibilité de configuration et en communication. De nos jours, ils sont entrain d'étendre leur utilisation en utilisant des processeurs numériques permettant des implémentations logicielles. Devant cette évolution rapide, on définira un système neuromorphique par une implémentation mixte à forte composante analogique, qui reproduit des structures et des fonctions neuronales.

# 1. Simulation matérielle des réseaux de neurones impulsionnels

Pour simuler des réseaux de neurones, une approche multidisciplinaire est généralement suivie. Les biologistes examinent les bases neurobiologiques de la structure et du fonctionnement du système nerveux. Les mathématiques interviennent dans un second temps pour définir des modèles qui reproduisent fidèlement les phénomènes observés. Des choix techniques et technologiques aboutiront à la réalisation des supports de simulation qui vont accueillir les modèles. Dans ce chapitre, on s'intéressera d'abord aux concepts liés à la simulation des réseaux de neurones d'une façon générale (section 1). Deux méthodes de simulations, à savoir simulation logicielle et simulation matérielle, sont présentées. A partir d'une comparaison des deux méthodes, on discutera les motivations qui amènent à établir des supports matériels pour modéliser et simuler des réseaux de neurones. La section 2 introduira les notions de base de la neurobiologie nécessaires pour la compréhension de ces travaux. Une sélection de modèles neuronaux appropriés à la compréhension des systèmes neuromorphiques présentés dans cette thèse est fournie dans la section 3. Un état de l'art des approches de simulation matérielle les plus utilisées dans le domaine de recherche scientifique termine ce chapitre et permet de positionner notre approche parmi d'autres.

## 1 Neurosciences et ingénierie neuromorphique

### 1.1 Les neurosciences modernes

La coopération de plusieurs disciplines scientifiques a donné naissance à des techniques et des approches nouvelles pour l'étude du système nerveux. Cette section passe en revue quelques techniques et avancées scientifiques qui ont dressé l'image moderne des neurosciences.

Pour comprendre les principes de fonctionnement du cerveau, deux aspects du système nerveux doivent être analysés : l'aspect morphologique, qui représente le substrat physique, et l'aspect codage neuronal qui est utilisé pour représenter, traiter et stocker l'information sous forme de patterns spatio-temporels (Gerstner, 2002).

Le début de l'étude de l'aspect morphologique a été marqué par Ramon y Cajal (Ramon y Cajal, 1911) qui a révélé la structure d'une cellule unique et d'un réseau de cellules dans leur milieu naturel. Depuis lors, les techniques de visualisation microscopique se sont améliorées considérablement. Il est même possible de nos jours d'identifier automatiquement un axone ou une dendrite et de tracer son volume à l'aide d'un microscope électronique (Briggman,

2006). Cependant, la morphologie complète d'un cerveau d'un vertébré n'est pas encore systématiquement effectuée.

D'un autre côté, la mesure de l'activité neuronale d'une culture *in-vitro* ou d'un système nerveux *in-vivo* est abordée par plusieurs techniques. La première technique non-invasive est l'électroencéphalographie (EEG) (Jung, 1979) qui est aujourd'hui accompagnée par la magnétoencéphalographie (MEG) (Cohen, 1968) et l'imagerie par résonnance magnétique (fMRI) (Logothetis, 2001). Les deux premières techniques font partie des mesures de surface, alors que la troisième permet des enregistrements plus en profondeur. La résolution spatiale de fMRI est de l'ordre du millimètre. Elle est considérée comme la meilleure résolution possible. La résolution temporelle de fMRI est de l'ordre de 2 secondes, tandis que l'EEG et la MEG peuvent mesurer une activité neuronale à l'échelle de la milliseconde. Une autre technique pour acquérir des images tridimensionnelles de l'activité des régions du cerveau est l'émission tomographique de positron, avec une résolution spatiale de l'ordre du millimètre et un taux d'échantillonnage arrivant jusqu'à 60 échantillons/s (Purschke, 2005).

Les méthodes invasives fournissent des résolutions spatiale et temporelle élevées, mais elles ne permettent d'accéder qu'à une petite portion du tissu neuronal. Les enregistrements intracellulaires utilisant les techniques de *patch-clamp* offrent une haute résolution en courant et en tension pour une membrane de cellule isolée (Sakmann, 1995). Les mêmes techniques permettent également de détecter la corrélation entre les fluctuations des potentiels de membranes de quelques cellules cultivées ensemble (Lampl, 1999). Ainsi, l'enregistrement de l'activité d'une paire ou d'un groupe de neurones peut révéler l'existence de connexions synaptiques et l'effet de la plasticité synaptique. Des changements à long terme dans l'efficacité synaptique ont été observés en corrélant l'activité présynaptique et postsynaptique et en enregistrant le potentiel de membrane (Bi, 1999). Des techniques similaires mettent l'accent sur les changements à court terme des effets de la plasticité (Markram, 1998).

Malgré ces méthodes d'investigation modernes, plusieurs champs restent à explorer : le codage de l'information dans le cerveau, la capacité d'apprentissage, la robustesse des fonctionnalités, la mémorisation, l'émergence de la créativité, etc., pour n'en citer que quelques unes.

### 1.2 L'apport de la modélisation

L'investigation expérimentale du tissu neuronal a un rôle primordial dans l'élaboration d'une base de connaissances permettant de mieux comprendre les mécanismes neuronaux qui régissent le cerveau. Cependant, plusieurs difficultés limitent le rôle de l'approche expérimentale. En voici quelques exemples :

- les contraintes technologiques sont lourdes,
- il est difficile de maîtriser les expériences *in-vivo*,
- les préparations *in-vitro* ne reflètent pas l'état naturel des cellules,
- il est difficile de tirer une conclusion à partir des analyses statistiques,
- l'interface artificiel-vivant induit des variabilités dans l'enregistrement,
- le coût de certaines expériences,
- les contraintes et régulations éthiques,…

Les modèles sont des instruments scientifiques essentiels à la description d'un problème spécifique à un contexte. En effet, modéliser un élément pour qu'il réagisse d'une manière semblable à celui qu'on veut étudier est une solution pour remplacer des expériences

neurobiologiques irréalisables, trop coûteuses, ou soumises à des considérations d'éthique. La construction d'un modèle combine des éléments de connaissances, des questions bien définies, des méthodes et des hypothèses pour extraire des comportements et ses prédictions cachées. En partant des résultats d'expérimentations biologiques, la modélisation pourra réalimenter en retour l'expérience biologique en orientant la démarche de l'expérimentation.

### 1.3 Simulation logicielle vs. Simulation matérielle

Les modèles des neurones impulsionnels sont formulés par des ensembles d'équations différentielles pour un traitement analytique ou une simulation numérique. Par rapport aux expérimentations biologiques, la simulation logicielle des modèles neuronaux offre une plus grande souplesse. Les modèles sont flexibles dans leur structure, les niveaux de détail et les choix des paramètres. L'usage des langages et des formats communs permet de plus un brassage et une mutualisation des efforts. Une large communauté scientifique qui utilise cette approche apporte des contributions significatives au domaine des neurosciences computationnelles. Par exemple, la variation des propriétés intégratives des membranes d'un neurone a été reproduite et expliquée par des modèles numériquement simulables (Shelley, 2002 ; Destexhe, 2003). Des modèles allant d'une description très détaillée au niveau d'une cellule unique (Jolivet, 2008) jusqu'à la description de réseaux à grande taille (Johansson and Lansner, 2007) ont été également simulés. Les découvertes qui en découlent peuvent alimenter en retour les expérimentations biologiques, ou garder leur aspect abstrait dans une tentative de compréhension des principes neuronaux.

Plus le modèle de neurone est complexe et plus la taille du réseau est grande, plus le temps de simulation devient critique. Des observations à long terme de la dynamique d'un réseau de neurones peuvent engendrer des calculs extrêmement coûteux (Morisson, 2007). Le goulet d'étranglement vient principalement des tentatives d'adaptation du parallélisme intrinsèque au calcul neuronal à des architectures essentiellement à processeurs séquentiels. Il est difficile de parler d'une simulation temps réel dans ces conditions.

Les systèmes neuromorphiques représentent une alternative pour surmonter quelques limitations inhérentes au calcul logiciel (Renaud, 2007). Leur point fort réside dans la capacité des composants électroniques à reproduire la structure et le fonctionnement des réseaux neuronaux biologiques : ils sont capables d'adopter les propriétés originales des paradigmes des réseaux neuronaux. La rapidité et la compacité des circuits VLSI rendent ces systèmes attractifs pour des implémentations massivement parallèles. D'autre part, les circuits analogiques opèrent dans un régime temps continu, identique à l'évolution de l'information dans les systèmes biologiques.

L'avantage principal de l'utilisation des systèmes neuromorphiques, en comparaison avec les supports de simulation numériques, surgit du fait que la simulation matérielle peut se faire en temps réel biologique ou plusieurs fois plus rapide. Le système réalisé dans le cadre de cette thèse fonctionne en temps réel biologique. Autrement dit, l'échelle temporelle de l'activité électrique dans les cellules vivantes est identique à celle de l'activité électrique dans les circuits et systèmes neuromorphiques. La majorité des systèmes neuromorphiques fonctionnent en temps réel biologique, pour des raisons d'interfaçage avec le monde réel. Cependant, d'autres systèmes exploitent la rapidité croissante des technologies VLSI pour réaliser des simulateurs de réseaux neuronaux opérant entre 1000 et 10000 fois plus rapide que le temps réel biologique (Schemmel, 2008). Le but est de fournir des outils de modélisation neuromorphique capables de compresser le calcul neuronal dans le temps, et donc simuler les mécanismes neuronaux qui s'opèrent à long terme.

Ayant toutes ces caractéristiques, les systèmes neuromorphiques sont plus susceptibles de communiquer avec le monde réel. Les systèmes de traitement de vision (Merolla, 2006), le

contrôle de l'autorégulation des moteurs d'un robot (Lewis, 2000), les rétines artificielles (Lichtsteiner, 2007) et bien d'autres applications technologiques en sont de bons exemples. Cependant, un contact plus immédiat avec les neurosciences est généralement exprimé par le couplage des instruments électroniques avec des tissus vivants (Bontorin, 2007 ; Nam, 2009), ou encore par le développement des outils de modélisation neuromorphique à des fins biologiques ou médicales (Renaud, 2007 ; Schemmel, 2007).

Les circuits et systèmes neuromorphiques sont sensibles au bruit, au cross-talk et à la température (Dally, 1998). Deux circuits issus d'une même fabrication ne fonctionneront jamais de la même façon. Ceci peut engendrer une calibration pour chaque composant analogique.

Contrairement à la simulation logicielle, la flexibilité des circuits neuromorphiques est limitée. Typiquement, les paramètres des neurones, des synapses et la topologie du réseau sont configurables, mais les modèles eux-mêmes ne le sont pas. Un changement mineur dans le modèle nécessite une re-conception suivie d'une phase de production et de test d'un nouveau circuit. Ce processus peut prendre plusieurs mois.

Une autre différence entre l'approche logicielle et l'approche matérielle découle de la dimension finie des supports de simulation. Dans la simulation logicielle la taille des réseaux est limitée par la mémoire et les ressources du processeur. Cette contrainte est plus immédiate dans les systèmes neuromorphiques : le nombre de neurones et le nombre de synapses par neurone sont limités. Cette limite est déterminée, dans un premier temps, pendant la phase de conception par la technologie utilisée et les ressources disponibles, et dans un second temps, par les contraintes temporelles et fonctionnelles relatives aux simulations.

Les deux approches ont clairement des avantages et des inconvénients techniques et technologiques qui favorisent une approche sur l'autre dans certains cas. Cependant, la simulation temps-réel des réseaux de neurones de tailles importantes trace les limites définitives de l'approche logicielle. Prenons l'exemple de la simulation d'un réseau contenant un million de neurones. La simulation logicielle nécessite un nombre important de processeurs qui coopèrent ensemble. On parle de baies de processeurs utilisés exclusivement pour cette tâche. La consommation de l'électricité peut atteindre plusieurs kilowatts par simulation. En plus de leur indisponibilité pour toute la communauté, ces ressources constituent un vrai défi financier pour les utilisateurs.

D'un autre côté, le temps réel biologique est très difficile à atteindre en utilisant l'approche logicielle. Markram, un des utilisateurs du supercalculateur *BlueGene* d'IBM pour simuler des réseaux de neurones biologiquement réalistes, a déclaré qu'il était capable de simuler seulement 50.000 neurones en temps réel biologique. La simulation de 1 million de neurones prendra entre 8 et 10 fois plus de temps que le temps réel biologique.

Par contre, le même nombre de neurones (1 million) est simulable en temps-réel à l'aide d'un système neuromorphique. Motivé par la capacité d'intégration des circuits électroniques qui ne cesse d'augmenter et des techniques avancées d'assemblage de plusieurs puces, ces systèmes deviennent souhaitables pour les simulations des réseaux de neurones de grande taille. En plus, quelques travaux proposent de garder les circuits sur la même galette de silicium et d'établir en post-fabrication les connexions entre elles (Schemmel, 2007). Ceci permettra de rassembler un nombre important de circuits et d'avoir des communications plus rapides.

Plusieurs systèmes neuromorphiques permettant une simulation à grande échelle sont dans leur phase finale de réalisation. L'équipe de l'université *Heidelberg* essaye d'intégrer un million de neurones *integrate-and-fire* sur des systèmes multipuces. Un autre système, appelé

*Neurogrid*, est constitué d'une grille de 16 puces contenant chacune une matrice de 256 x 256 neurones de type Hodgkin-Huxley (Boahen, 2006). La consommation d'énergie dans ces systèmes est de l'ordre de quelques Watts. Ces travaux rendent la simulation matérielle attractive pour le calcul neuronal à grande échelle.

### 1.4 Les approches neuromorphiques : un nouveau paradigme de calcul

*Les caractéristiques du calcul neuromorphique*

Le calcul neuromorphique utilise une collection massive de synapses et de neurones qui fonctionnent en parallèle d'une façon asynchrone. Les neurones communiquent entre eux pour assurer une fonction donnée ou résoudre un problème. Cette architecture adoptée par les systèmes neuromorphiques possède un certains nombre de caractéristiques qui la différencient d'autres architectures de calcul ; on peut citer :

- le calcul parallèle,
- la communication un-à-plusieurs entre les neurones,
- la coopération de plusieurs éléments pour assurer une fonction spécifique,
- un traitement indéterministe mais robuste,
- le traitement basé sur l'expérience,
- l'apprentissage, l'adaptation et l'interaction avec l'environnement externe,
- l'évolution en temps réel,
- la faible utilisation d'énergie.

Un système possédant ces capacités de calcul peut être utilisé dans de nombreux domaines. Il peut piloter un robot, réaliser des optimisations, reproduire des fonctions cognitives du cerveau, auto-évaluer et auto-évoluer, mémoriser des données, déduire des conclusions, prendre des décisions, etc. Ces fonctions sont fort probables les bases d'un nouvel paradigme de calcul et des nouvelles technologies.

Les concepteurs de systèmes neuromorphiques tendent à reproduire ces caractéristiques dans leurs systèmes. Certains se contentent de l'aspect comportemental des mécanismes neuronaux dans leurs réalisations, alors que d'autres creusent dans les aspects biophysiques pour s'approcher plus du biologique. Quelque soit l'approche suivie, la question sur la faisabilité de côtoyer la capacité du calcul du cerveau reste toujours ouverte. Est-ce que la technologie existante, ou future, permettra de développer des systèmes qui simulent un cerveau ?

*Vers des systèmes à l'échelle du cerveau*

Le nombre important d'opérations qui se déroulent en parallèle dans le cerveau est très grand. Il est estimé à $10^{16}$ opérations/s dans une étude statistique réalisée par C. Mead (Mead, 1990). Bien que les technologies récentes des circuits intégrés ont beaucoup évolué en termes de rapidité de calcul et capacité d'intégration, ils restent, cependant, très loin d'atteindre les performances du cerveau.

De plus, le cerveau utilise seulement 20 Watts pour réaliser des fonctions cognitives complexes. En moyenne, il utilise $10^6$ J pour réaliser une opération (Mead, 1990). En comparant ce chiffre avec l'énergie utilisée dans les simulateurs matériels les plus performants que l'on puisse imaginer, on trouve que le cerveau utilise $10^7$ fois moins d'énergie. La réalisation des simulations à grande échelle peut engendrer un coût prohibitif pour sa mise en ouvre. Ce problème est déjà effectif dans le domaine de la robotique.

La solution pourrait venir de deux sources : l'optimisation des modèles élémentaires et le développement de nouvelles techniques de fabrication. L'optimisation des modèles consiste à concilier la dynamique avec la simplicité ; il s'agit d'appliquer des abstractions qui simplifient le fonctionnement de base tout en gardant l'information pertinente.

Les solutions qui peuvent provenir des techniques de fabrication sont des alternatives d'implémentation directes des modèles. Autrement dit, l'élément de base des circuits électroniques devient lui-même l'élément de base des circuits neuromorphiques. Ainsi, le modèle d'un transistor peut reproduire le comportement d'un neurone ou d'une synapse. Cet axe de recherche devient de plus en plus intéressant pour les concepteurs des structures nanométriques. Un exemple très récent de la réalisation d'un transistor qui se comporte comme une synapse biologique se trouve dans (Alibart, 2010). Les auteurs exploitent la physique des semi-conducteurs en analogie avec la biologie pour réaliser des fonctions d'apprentissage.

Une bonne optimisation des modèles accompagnée d'une réalisation efficace permet une intégration compacte et un fonctionnement fidèle. Le nombre de composants neuronaux que l'on peut intégrer sur un circuit électronique augmente considérablement. Des simulations à grande échelle deviennent donc possibles. Cependant, il n'est pas prévu que l'on peut atteindre l'échelle du cerveau avec la technologie silicium. D'autres technologies, dites non-silicium, sont prévues d'être capables d'accueillir des réalisations à l'échelle du cerveau ; la technologie organique en est un bon exemple. Ceci dit, de telles réalisations sont prévues dans une cinquantaine d'années.

## 2 Les bases de neurobiologie

La présentation des notions biologiques dans cette section est non exhaustive et hautement simplifiée. Pour une présentation plus détaillée, le lecteur est invité à consulter la littérature mentionnée à la fin du manuscrit. Néanmoins, on essayera de fournir un minimum d'informations nécessaires permettant au lecteur d'apprécier les notions neurobiologiques des modèles théoriques utilisés dans le cadre de cette thèse.

### 2.1 Les éléments d'un système neuronal

Depuis les années 1900, une grande quantité de travaux de recherche en biologie ont apporté des connaissances nouvelles sur la structure et la fonction des systèmes neuronaux. Les éléments responsables de la production de l'information neuronale sont les neurones, ou cellules nerveuses. Les neurones sont interconnectés entre eux formant une structure volumique complexe. Une portion minuscule d'un réseau de neurones est illustrée dans la figure 1.1 qui montre une photographie à l'échelle microscopique d'un tissu nerveux d'un rat. On peut distinguer plusieurs formes du corps des neurones, triangulaires ou circulaires, et des extensions qui se terminent sur d'autres neurones. Cette image n'est qu'un aperçu de ce qu'est un réseau de neurones. En réalité, les neurones et leurs connexions sont tassés dans un réseau dense de $10^4$ cellules et de quelques kilomètres de ramifications par millimètre cube.

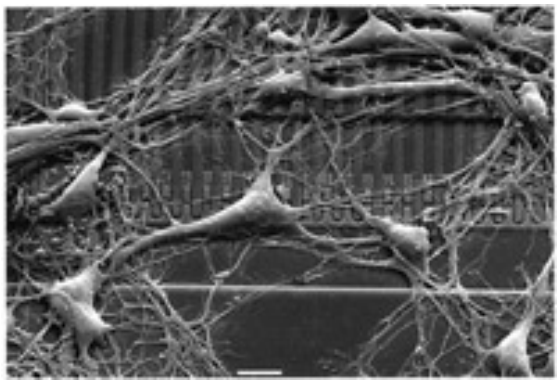


Figure 1.1 : Une photographie des cellules nerveuses d'une culture *in-vitro* du néocortex d'un rat.

Le cortex cérébral est, à l'instar de tous les cortex de l'organisme, la partie périphérique des hémisphères cérébraux. Il est supposé être le siège des fonctions neurologiques élaborées. Les physiologistes ont proposé une subdivision en couches du cortex. Le nombre de couches change selon les espèces. Les couches sont numérotées depuis la surface. Par exemple, dans le cortex humain on identifie six couches. Ces couches corticales ne sont pas un simple empilement de neurones. Les neurones s'organisent en unités fonctionnelles prenant la forme de colonnes perpendiculaires à la surface du cortex, chacune assurant une fonction. Il s'agit des zones fonctionnelles, appelées aussi *aires*, qui assurent des fonctions cognitives. On distingue trois grands types de zones : les aires sensorielles, les aires motrices et les aires d'association (Mountcastle, 1997).

Le cortex ne contient pas exclusivement des neurones. Il y a d'autres cellules, appelées *glia*, qui assurent l'apport énergétique et la stabilisation structurelle du tissu nerveux. Ces glia n'interviennent pas dans le traitement de l'information neuronale, raison pour laquelle on ne s'intéressera pas à elles dans ce manuscrit.

### 2.2 Le neurone impulsionnel

Un neurone peut être divisé en 3 parties fonctionnelles : le soma, l'axone et les dendrites. Les dendrites sont les dispositifs récepteurs du neurone. Elles collectent les signaux venant des autres neurones et les dirigent vers le soma. Le soma est le centre du traitement de l'information neuronale. Il effectue un traitement non-linéaire : si le total des signaux dendritiques entrant dépasse un certain seuil, alors un signal de sortie est généré. Ce dernier est véhiculé par le dispositif de sortie du neurone, l'axone, vers les autres neurones. L'axone peut arriver tout près des neurones de son voisinage, comme il peut s'étendre sur plusieurs centimètres et atteindre d'autres régions du cerveau. La jonction qui sépare les terminaisons de l'axone et les autres neurones est appelée synapse. Dans le cortex des vertébrés, un seul neurone peut avoir $10^4$ synapses. La figure 1.2 illustre les trois parties d'un neurone à travers une microphotographie d'une cellule.

Supposons qu'un neurone émette un signal qui traverse une synapse pour atteindre un deuxième neurone ; par convention, le neurone qui a généré le signal est dit neurone présynaptique, et le neurone récepteur est dit neurone postsynaptique.

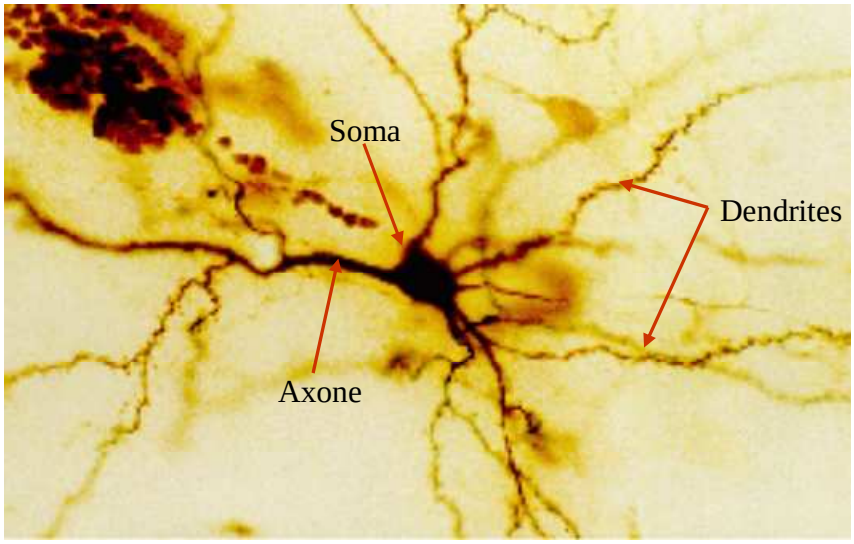



Figure 1.2 Microphotographie d'un neurone.

Dans le signal électrique neuronal peuvent être identifiées des impulsions électriques de courte durée. Pour observer ce signal, il faut placer une électrode près du soma ou de l'axone. Ces impulsions, appelées aussi potentiels d'action, ont une amplitude d'environ 100 mV et une durée typique de 1 à 5 ms. La forme d'un potentiel d'action change peu. Elle est la même tout au long de l'axone. L'essentiel de l'information neuronale n'est pas codée par la forme du potentiel d'action mais plutôt par leur nombre, leur rythme et le moment de génération. Le potentiel d'action est l'unité élémentaire d'une transmission neuronale. Les neurones qui échangent les informations entre eux par l'intermédiaire des potentiels d'actions sont appelés neurones impulsionnels.

Le potentiel d'action est le résultat d'une dépolarisation transitoire, locale, brève et stéréotypée de la membrane du neurone. Un deuxième potentiel d'action ne peut être généré tant que la membrane cellulaire n'est pas suffisamment repolarisée. On parle de période réfractaire. Cette période diffère d'un type de neurone à un autre. Elle varie de 1 à 15 ms dans les cellules corticales. On utilisera par la suite cette grandeur pour définir le taux d'émission maximal pour des modèles de neurones.

### 2.3 La synapse

Le lieu où la terminaison de l'axone du neurone présynaptique entre en contact avec la dendrite ou le soma du neurone postsynaptique est la synapse. Le plus souvent, on rencontre des synapses dites chimiques : quand un potentiel d'action arrive à une synapse, il déclenche une série de réactions biochimiques conduisant à la libération des neurotransmetteurs de la terminaison présynaptique dans la jonction synaptique. Dès que les neurotransmetteurs arrivent du côté postsynaptique, ils sont détectés par des récepteurs membranaires postsynaptiques qui ouvrent des canaux ioniques spécifiques de telle sorte que les ions peuvent pénétrer du milieu extracellulaire dans le milieu intracellulaire. Le flux ionique, à son tour, conduit au changement du potentiel de la membrane postsynaptique. Les réactions chimiques induites se transforment en une réponse électrique.

A part les synapses chimiques, les neurones peuvent aussi être couplés par l'intermédiaire des synapses électriques. Des potentiels de membranes spécifiques font une connexion électrique résistive entre les deux neurones. Ce type de synapse est susceptible d'intervenir au niveau de la synchronisation entre les neurones.

Les synapses possèdent deux effets antagonistes. Elles peuvent être excitatrices pour le neurone postsynaptique ou au contraire inhibitrices. Chez les vertébrés, l'effet de l'inhibition fait intervenir des neurotransmetteurs appelés acide γ-aminobutyrique ou GABA. L'effet excitateur, quant à lui, est généralement le résultat des réactions chimiques déclenchées par les neurotransmetteurs glutamate.

Le nombre de synapse dans le cortex humain est estimé à $10^{15}$ synapses. Le diamètre d'un contact synaptique est approximativement de 1 µm (Shepherd, 2004). Cette densité structurelle rend la surveillance des phénomènes physiologiques très difficile, ce qui ajoute un obstacle supplémentaire sur le chemin de la compréhension de la relation entre les structures et les fonctions.

### 2.4 La plasticité synaptique

Les changements qui peuvent faire varier la force d'une connexion synaptique définissent la plasticité synaptique. Les expériences biologiques qui mettent en évidence les changements au niveau de la synapse débutent vers les années 1970. L'expérience de base était de placer une électrode pour stimuler un neurone présynaptique et une deuxième pour enregistrer la réponse du neurone postsynaptique (Bliss, 1973). En faisant varier l'intensité et la fréquence des stimuli, des changements durables affectent l'efficacité synaptique. Les résultats obtenus ont permis de conclure que la force de la synapse est une fonction de l'activité pré et postsynaptique. Les changements engendrés sont connus sous le nom de LTP (pour *Long-Term Potentiation*) lorsque l'efficacité de la synapse augmente, et LTD (pour *Long-Term Depression*) dans le cas contraire.

Beaucoup d'expérimentations ont été inspirées par le postulat de Hebb (Hebb, 1949) selon lequel une coïncidence d'activité entre deux neurones couplés par l'intermédiaire d'une synapse pourrait provoquer une augmentation de l'efficacité de la transmission synaptique entre ces deux neurones.

Hebb a formulé son principe sur des bases purement théoriques. C'est seulement 20 ans plus tard qu'il a été validé expérimentalement (Bliss, 1973). A l'issue des résultats expérimentaux, le postulat de Hebb est reformulé avec des mots plus appropriés : les modifications dans l'efficacité de la transmission synaptique sont régies par des corrélations entre séquences d'activité des neurones pré et postsynaptique.

Avec des techniques d'enregistrement intracellulaires modernes, il est devenu possible de stimuler et enregistrer un ou plusieurs neurones avec une excellente résolution spatiale et temporelle (Magee, 1997). Les résultats qui en découlent mettent l'accent sur l'aspect temporel et révèlent que l'efficacité synaptique évolue également en fonction de la différence des temps de génération des potentiels d'action des neurones pré et postsynaptique. Cette nouvelle constatation ajoute l'aspect temporel aux interprétations des mécanismes de plasticité. La figure 1.3 illustre des expériences pionnières réalisées sur des paires de neurones où ils sont forcés à générer des potentiels d'action à des temps prédéterminés (Bi et Poo, 1998). Une nouvelle image de la plasticité synaptique émerge : la plasticité dépendante des instants des potentiels d'action.

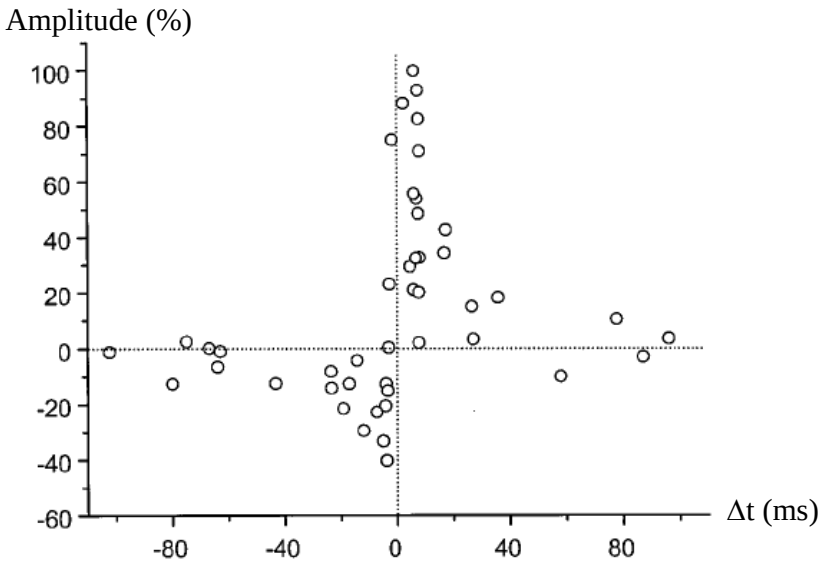



Figure 1.3 Evolution de l'efficacité synaptique en fonction des instants des potentiels d'action pré et postsynaptique. Δt est la différence entre les temps de génération des potentiels d'action pré et postsynaptique. Lorsque le potentiel d'action présynaptique précède le potentiel d'action postsynaptique, l'amplitude du changement synaptique augmente. Dans le cas contraire, elle diminue. La grandeur des modifications est déterminée par la fenêtre de temps séparant les deux potentiels d'action (Bi and Poo, 1998).

Certes, la découverte de l'aspect temporel du changement synaptique a clarifié la fonction de la plasticité synaptique, mais elle ne conclut pas sur le mécanisme de plasticité. En effet, des formes de plasticité, dites *anti-Hebbiennes*, ont pu être observées (Bell, 1997) ; l'efficacité synaptique change d'une façon globalement antagoniste à celle illustrée dans la figure 1.3. Parallèlement, il a été montré que les paramètres de la plasticité (amplitude de changement, fenêtre de dépression) dépendent de la localisation dendritique (Fromke, 2005).

## 3 Les modèles utilisés

Cette section passe en revue quelques modèles mathématiques qui reproduisent les phénomènes neurophysiologiques décrits dans la section précédente. Seuls les modèles qui ont une relation directe ou indirecte avec les travaux de cette thèse seront présentés. On s'intéressera en particulier aux modèles de neurone, synapse et plasticité de réseau. Combinés avec la connectivité, ces modèles définiront des réseaux de neurones impulsionnels de propriétés différentes.

### 3.1 La modélisation du soma

La pertinence d'un modèle de neurone est reflétée par sa fidélité de comportement et sa simplicité d'implémentation. Pour du calcul logiciel ou matériel, il est cependant nécessaire de faire des compromis entre la complexité des modèles et leur facilité d'implémentation.

Les modèles de neurones présentés ici possèdent une abstraction commune : un neurone est représenté par un point dans l'espace. Ces modèles sont dédiés à une implémentation réseau, où seules les propriétés à l'échelle de réseau sont utiles. Ainsi, les phénomènes de propagation le long de la membrane, les distorsions dans l'arbre dendritique ou la déformation du potentiel d'action le long de l'axone ne seront pas pris en compte.

*Le modèle Integrate-and-fire*

Dans ce modèle de neurone, une simplification majeure est apportée à la forme des potentiels d'action. Chaque potentiel d'action est considéré comme un événement défini seulement par son instant d'apparition. Les phases de polarisation et de dépolarisation ne sont pas représentées.

Les contributions de chaque neurone présynaptique sont additionnées algébriquement à un potentiel de membrane initialisé arbitrairement. Lorsque le potentiel de membrane franchit un seuil préfixé, on considère que le neurone émet un potentiel d'action (Gerstner, 2002). Les stimulations afférentes sont ignorées pendant la période réfractaire, puis le potentiel de membrane est réinitialisé à sa valeur de départ.

Si on considère une implémentation électronique analogique, les stimulations afférentes sont transformées en quanta de courant injectés dans un condensateur. Un comparateur détecte si la charge du condensateur dépasse un seuil prédéterminé pour générer un évènement. Un circuit spécifique remet la charge à sa valeur initiale après la génération de l'évènement. La fréquence de génération des potentiels d'action dans un neurone augmente linéairement avec l'augmentation du courant de stimulation.

*Le modèle Integrate-and-fire avec fuite*

Un potentiel d'action est toujours considéré comme un évènement dont la forme importe peu. C'est la notion d'une évolution temporelle qui est ajoutée au modèle *Integrate-and-fire*. Dans ce modèle, on tient compte des courants de fuites qui reflètent la diffusion permanente d'ions à travers la membrane cellulaire. La diffusion d'ions se rapporte à un pompage actif qui tend à ramener en permanence le potentiel de membrane à son état d'équilibre ($E_{équi}$).

La dynamique de ce modèle est représentée par l'équation différentielle d'un circuit RC. Le modèle peut être décrit comme suit,

$$C_{mem}\frac{dV_{mem}}{dt} = I(t) - \frac{V_{mem}(t) - E_{équi}}{R} = I(t) - g_F.(V_{mem}(t) - E_{équi}) \qquad (1.1)$$

où $V_{mem}$ est le potentiel de membrane, *I(t)* le courant d'entrée issu des synapses et $g_F$ la conductance qui représente le courant de fuite ($g_F$ = 1/R). Le circuit de base comprend une capacité $C_{mem}$ branchée en parallèle avec une conductance $g_F$ comme le montre la figure 1.4. La notion de fuite est représentée par la conductance $g_F$. La source de tension qui produit $E_{équi}$ définit le potentiel d'équilibre. Lorsque $V_{mem} = E_{équi}$ le courant de fuite est nul.

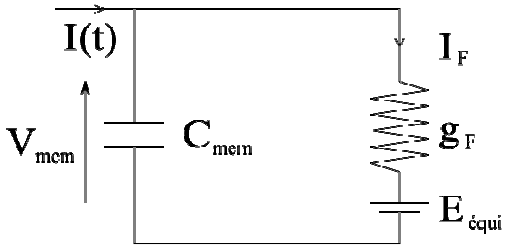



Figure 1.4 Mise en circuit du modèle *Integrate-and-Fire* avec fuite.

Le neurone émet un potentiel d'action à chaque fois que le $V_{mem}$ dépasse un certain seuil. Immédiatement après l'instant de l'émission, le potentiel de membrane est remis à zéro. Une période de réfraction peut être modélisée en forçant le potentiel de membrane à une valeur basse (ou même négative) durant une certaine période de temps.

Ce modèle est plus proche de la dynamique de la membrane cellulaire. Il offre un compromis intéressant entre la fidélité du comportement et la puissance de calcul. Raison pour laquelle il est très utilisé dans le cadre des neurosciences computationnelles pour la simulation des réseaux de moyenne et grande taille.

*Le formalisme de Hodgkin-Huxley*

Du point de vue de la biophysique, un potentiel d'action est le résultat des courants créés par le passage d'ions de part et d'autre de la membrane cellulaire par l'intermédiaire des canaux ioniques. En isolant un axone géant de calmar, les physiologistes Hodgkin et Huxley ont réussi à mesurer les courants ioniques et à décrire leur dynamique par le moyen d'équations différentielles. A partir de 1949, dans une série de publications regroupées et synthétisées en 1952, les deux physiologistes exposent un modèle qui est devenu une référence importante dans le domaine des neurosciences (Hodgkin et Huxley, 1952).

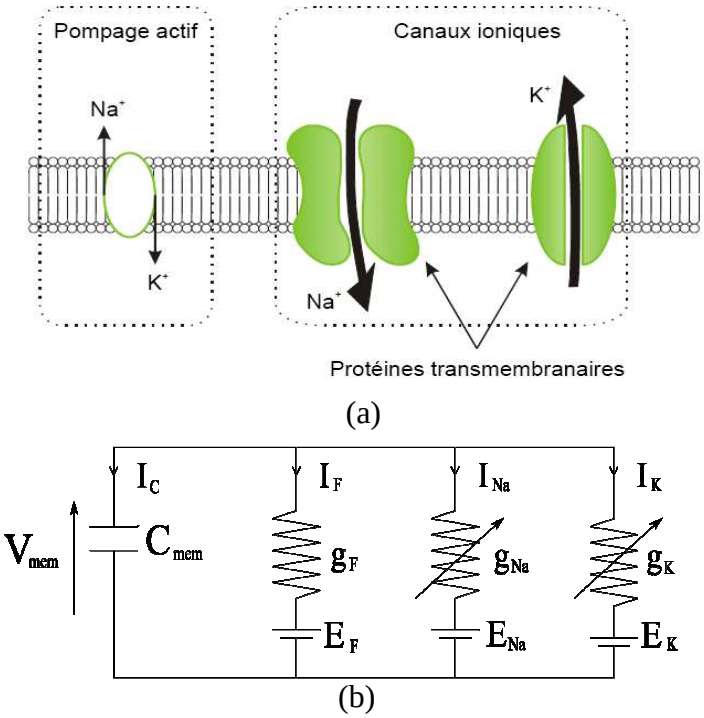



Figure 1.5 (a) Représentation des canaux ioniques de base présents sur une membrane de neurone (b) Le schéma électrique équivalent à la membrane cellulaire (modèle de Hodgkin-Huxley).

Dans leurs expérimentations, Hodgkin et Huxley mettent l'accent sur trois types de courants ioniques : les courants sodique, potassique et de fuite. Le modèle qu'ils ont proposé peut être décrit à l'aide de la figure 1.5. La membrane cellulaire semi-perméable sépare le milieu intracellulaire du milieu extracellulaire et agit comme une capacité. Le mouvement des ions actifs de part et d'autre de la membrane modifie la concentration en ions du milieu intracellulaire. Le potentiel de la membrane cellulaire suit cette variation représentée par la tension $V_{mem}$ appliquée aux bornes d'une capacité. Le courant $I_C$ est fonction des courants $I_K$, $I_{Na}$ et $I_F$ qui représentent respectivement les canaux potassiques, sodiques et de fuite. On a donc,

$$C_{mem}\frac{dV_{mem}}{dt} = -I_K - I_{Na} - I_F \qquad (1.2)$$

Dans leur modèle de base, Hodgkin-Huxley ne prennent en compte que ces trois courants ioniques. La perméabilité ionique de la membrane est représentée par les conductances ioniques $g_{Na}$, $g_K$ et $g_F$. La conductance du courant de fuite ne dépend pas du potentiel de la membrane, alors que les autres courants en dépendent.

$$\begin{cases} g_{Na}(V_{mem}) = \overline{g_{Na}}.m^3(V_{mem}).h(V_{mem}) \\ g_K(V_{mem}) = \overline{g_K}.n^4(V_{mem}) \end{cases} \quad (1.3)$$

où $V_{mem}$ représente le potentiel de membrane, $\overline{g_{Na}}$ et $\overline{g_K}$ représentent respectivement les valeurs maximales des conductances sodique et potassique. Les grandeurs m, n et h représentent la réponse temporelle à l'ouverture ou à la fermeture des canaux, m et n sont les termes d'activation (ouverture du canal), et h est un terme d'inactivation (fermeture du canal). Les variables m, n et h évoluent selon les équations différentielles suivantes,

$$\begin{cases} \dot{m} = \alpha_m(V_{mem})(1-m) - \beta_m(V_{mem})m \\ \dot{n} = \alpha_n(V_{mem})(1-n) - \beta_n(V_{mem})n \\ \dot{h} = \alpha_h(V_{mem})(1-h) - \beta_h(V_{mem})h \end{cases} \quad (1.4)$$

Les fonctions $\alpha$ et $\beta$ sont calculées pour modéliser les résultats de l'expérience sur l'axone géant de calmar.

En utilisant ce modèle de base, plusieurs dynamiques peuvent être reproduites, en allant d'une légère variation du potentiel de membrane à la génération des trains de potentiels d'action. L'ajout d'autres canaux ioniques, tel qu'un canal de calcium (*Ca*), apporte une dimension supplémentaire à la dynamique du modèle. Il devient, par exemple, possible d'observer les phénomènes d'adaptation (modulation) dans un train de potentiels d'action. De nombreuses dynamiques d'activité d'un neurone peuvent être reproduites avec ce modèle (Izhikevich, 2004). Cette flexibilité du modèle vient de la possibilité d'ajout de canaux ioniques qui reproduisent plusieurs états membranaires observés dans la biologie. Il permet, entre autres, de répondre à des questions liées à l'intégration synaptique, les effets morphologiques des dendrites, les enjeux entre les canaux ioniques et les dynamiques d'un neurone. Toute nouvelle contribution au modèle Hodgkin-Huxley se fait par l'ajout d'un ou de plusieurs canaux ioniques qui entrent en réaction avec les canaux ioniques de base. On obtient alors une représentation semblable à celle de la figure 1.6. Cette constatation nous amène à considérer une généralisation du modèle de Hodgkin-Huxley vers un *formalisme de Hodgkin-Huxley*.

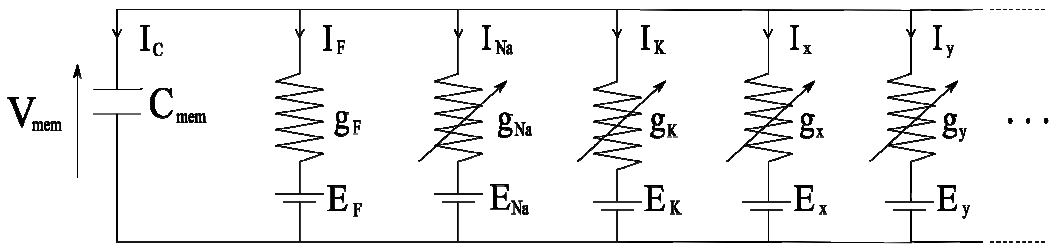



Figure 1.6 Le formalisme de Hodgkin-Huxley.

### 3.2 La synapse

*La monosynapse*

Une synapse utilise l'information neuronale transmise par le neurone présynaptique pour influer sur le potentiel de membrane postsynaptique. D'après le même formalisme que précédemment, modifier le potentiel de membrane revient à injecter un courant dans la capacité. Il s'agit donc d'un autre type de canal qui modélise la libération des neurotransmetteurs dans la jonction synaptique.

La modélisation du canal synaptique a été établie à partir d'enregistrements physiologiques et non pas d'une étude biophysique. Autrement dit, les modèles de la synapse reproduisent le phénomène synaptique et non pas le mécanisme biophysique qui engendre ce phénomène. L'intégration du courant synaptique se fait de la même façon que le formalisme des canaux ionique (équation 1.5),

$$\begin{cases} I_{syn}(t) = g_{syn}(t).(V_{mem} - E_{syn}) \\ g_{syn}(t) = \overline{g_{syn}}.r(t) \end{cases} \tag{1.5}$$

où $\overline{g_{syn}}$ est la conductance maximale du canal synaptique. La fonction *r(t)* modélise le temps de réponse de la synapse suite à l'ouverture des vésicules de neurotransmetteur. Le modèle de la synapse décroissante utilise une exponentielle décroissante dans la fonction *r(t)*. L'équation 1.6 et la figure 1.7 décrivent le comportement d'une synapse décroissante isolée (Abbott, 1994).

$$\begin{cases} r(t) = 0 & \forall t < t_0 \\ r(t) = W_{syn}.\exp(-\dfrac{t - t_0}{\tau_{syn}}) & \forall t \geq t_0 \end{cases} \tag{1.6}$$

où $\tau_{syn}$ représente la vitesse de décroissance du courant, $t_0$ l'instant auquel la stimulation doit être envoyée et $W_{syn}$ le poids de la synapse considérée. Le signe de $W_{syn}$ détermine le type de la synapse : excitatrice ou inhibitrice.

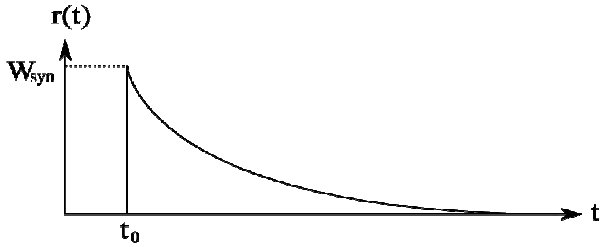



Figure 1.7. Evolution de la synapse en exponentielle décroissante suite à une stimulation produite à $t_0$ selon l'équation (1.6)

*La multisynapse*

Le modèle de synapse que l'on vient de décrire ne représente qu'une seule connexion entre neurones. En biologie, un neurone peut avoir jusqu'à $10^4$ connexions, soit $10^4$ synapses (même si la majorité sont peu efficaces). Il devient évident que ce modèle de synapse ne se prête pas à une implémentation niveau réseau. Pour connecter les N neurones d'un réseau entre eux en utilisant le modèle de monosynapse, on a besoin de N-1 monosynapses par neurone. En termes de canaux ioniques, on rajoute à chaque neurone N-1 canaux ioniques en

plus de ses canaux ioniques qui modélisent le soma. Cette solution est très coûteuse en termes de ressources matérielles et n'est possible que dans les réseaux à petites tailles.

Un autre modèle propose de combiner les effets de toutes les synapses afférentes dans un seul canal synaptique. Il repose sur une sommation des exponentielles décroissantes de chaque synapse. Les stimulations produites sont cumulées dans un même courant synaptique qui sera véhiculé par un seul canal synaptique, le canal de la multisynapse (Destexhe, 1999). L'équation 1.7 fournit la fonction $g_{syn}$ pour n stimulations se produisant aux instants $t_i$ avec i = 1, …, n.

$$g_{syn}(t) = \overline{g_{syn}} . \sum_{i=1}^{n} r_i(t) \qquad (1.7)$$

En utilisant la même fonction *r(t)* du modèle de la synapse décroissante, on aura une somme d'exponentielles décroissantes à calculer par multisynapse. La parade à ces exigences de calcul est d'utiliser une propriété de la fonction exponentielle : quelles que soient les valeurs *A*, *B*, $t_a$ et $t_b$ réels, il existe *C* réel qui vérifie la formule 1.8.

$$A.\exp(\frac{t-t_a}{\tau}) + B.\exp(\frac{t-t_b}{\tau}) = C.\exp(\frac{t-t_b}{\tau}) \qquad \text{pour } t > t_b \qquad (1.8)$$

Ce qui signifie que toutes les synapses ayant la même constante de temps τ peuvent être regroupées dans une même fonction *r(t)*. La figure 1.8 représente cet effet. Ce calcul est très utile pour les neurones qui possèdent des synapses modélisées par les mêmes équations.

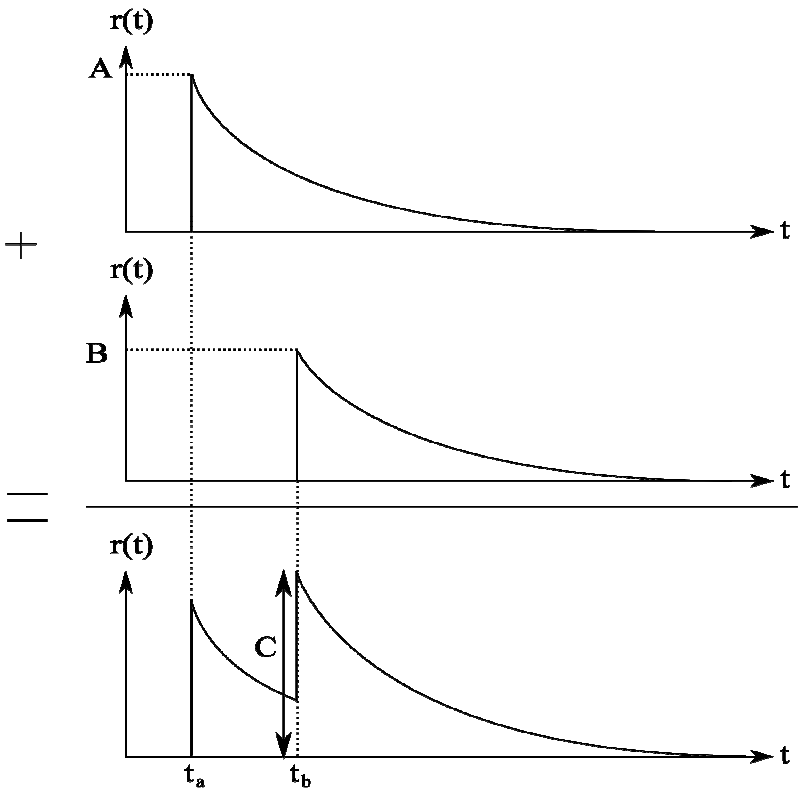



Figure 1.8. Sommation des fonctions *r(t)* d'un modèle de synapse décroissante. Toutes les courbes sont des exponentielles décroissantes de même constante de temps. La troisième courbe est la somme des deux premières selon l'équation 1.8.

L'inconvénient majeur du modèle de la multisynapse vient du fait que toutes les stimulations contribuent ensemble à construire le courant synaptique. Il ne devient plus possible de déterminer l'effet individuel de chaque stimulation sur le potentiel de membrane.

En parallèle, selon le modèle de l'équation 1.7, la valeur de r(t) peut évoluer vers des grandes valeurs. Cette constatation a un effet direct sur l'augmentation de l'amplitude du courant synaptique. Un phénomène de saturation intervient et limite l'augmentation du courant. La saturation s'exprime par une modulation de $W_{syn}$ qui peut se traduire par l'équation 1.9.

$$w_{syn}(t) = W_{syn}(t).\frac{\overline{g_{syn}} - g_{syn}(t)}{\overline{g_{syn}}} \tag{1.9}$$

### 3.3 La plasticité (STDP)

Dans cette section, on décrira le modèle de plasticité utilisé dans le cadre de cette thèse. Le modèle appartient à la famille des modèles dépendant des instants de génération des potentiels d'action, ou encore STDP pour *Spike-Timing-Dependent Plasticity*. D'une façon générale, la fluctuation des poids synaptiques dans les modèles STDP a pour origine l'ordre de succession des potentiels d'action pré et postsynaptiques dans une paire de neurones isolés. Si un neurone présynaptique émet un potentiel d'action avant le neurone postsynaptique, il contribue au déclenchement du second. Le poids synaptique est alors augmenté. Ce phénomène est appelé potentiation ou LTP (*Long-Term Potentiation*). Par contre, si l'ordre de succession des potentiels d'action se déroule dans l'ordre inverse, le poids synaptique décroît. Il s'agit alors d'une dépression ou LTD (*Long-term Despression*). L'amplitude des variations est d'autant plus faible que l'écart temporel entre les potentiels d'action est élevé. La figure 1.9 illustre le fonctionnement d'un modèle STDP inspiré des expérimentations biologiques (Destexhe, 2004).

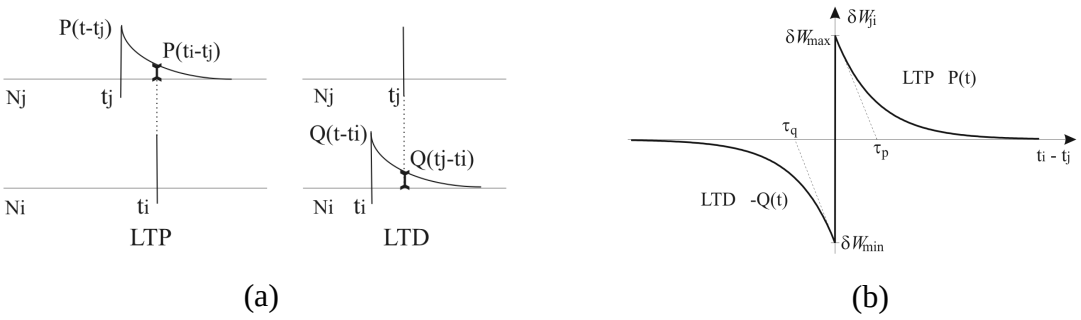



Figure 1.9 (a) *Nj* et *Ni* représentent respectivement les neurones pré et postsynaptique. Une potentiation (LTP) est observée lorsqu'un potentiel d'action présynaptique précède un potentiel d'action postsynaptique. Une dépression (LTD) est observée dans le cas contraire. (b) L'amplitude des variations du poids synaptique dû aux phénomènes de LTP (la fonction P(t)) et LTD (la fonction Q(t)) est une fonction de l'intervalle de temps qui sépare les potentiels d'actions pré et postsynaptiques.

On remarque dans ce modèle qu'il y a une non-linéarité lorsque les potentiels d'action arrivent simultanément. Les variations sont extrêmes, mais le signe de la contribution ne peut pas être déterminé à cause de l'incertitude autour de la notion de simultanéité. Ce problème disparait en forçant un ordre de détection de potentiels d'action. De cette façon, le signe de la variation dépendra des instants d'apparition des potentiels d'action.

Dans ce qui suit, on décrira l'équation qui représente le modèle utilisé. En plus de la potentiation et la dépression, le modèle comporte d'autres dépendances temporelles

complexes entre les instants des potentiels d'actions pré et postsynaptiques. Le but est de s'approcher le plus possible des résultats expérimentaux.

*Les effets mémoires*

On commencera par souligner le modèle général dans lequel l'amplitude et le signe du changement plastique sont déterminés par deux fonctions ″mémoire″, P et Q, qui codent le temps et l'amplitude de l'influence mutuelle entre les potentiels d'action pré et postsynaptiques. Selon le modèle, la variation du poids synaptique est donnée par l'équation (1.10) suivante

$$\frac{dw_{ij}}{dt} = \sum_{t_i} P\left[(t - t_j^{last}(t)\right]\delta(t - t_i) - \sum_{t_j} Q\left[(t - t_i^{last}(t)\right]\delta(t - t_j) \qquad (1.10)$$

$$avec \quad P(t) = A_p \exp(-t/\tau_p) \quad and \quad Q(t) = A_q \, exp(-t/\tau_q)$$

*P* et *Q* sont les fonctions qui décrivent respectivement l'amplitude de la potentiation (LTP) et de la dépression (LTD) au cours du temps. Elles sont représentées par des fonctions exponentielles décroissantes (figure 1.9). $\tau_p$ et $\tau_q$ sont les constantes de temps, tandis que $A_p$ et $A_q$ représentent respectivement les amplitudes initiales des fonctions *P* et *Q*. Les amplitudes de LTP et LTD dépendent de l'intervalle de temps relatif entre les deux potentiels d'action. La somme dans l'équation (1.10) couvre tous les potentiels d'action pré et postsynaptiques de telle sorte qu'à chaque fois qu'un potentiel d'action arrive, il interagit avec ces antécédents. Les impulsions de Dirac sélectionnent ou désélectionnent les fonctions, P ou Q, selon la succession des potentiels d'action.

*Efficacité des potentiels d'action*

La formule (1.10) suppose implicitement que l'interaction entre le neurone présynaptique et le neurone postsynaptique suit une loi de superposition linéaire des changements synaptiques. Cependant, des observations biologiques indiquent que les interactions ne sont pas linéaires et que la première paire de potentiels d'action tend toujours à primer sur les paires suivantes (Sjöström, 2001 ; Froemke, 2002). Pour modéliser ces effets, l'équation (1.10) est enrichie par des fonctions de *suppression*, représenté par les fonctions $\varepsilon_j$ et $\varepsilon_i$,

$$\frac{dw_{ij}}{dt} = \varepsilon_i \; \varepsilon_j \; \{ \sum_{t_i} P\left[(t - t_j^{last}(t)\right]\delta(t - t_i) - \sum_{t_j} Q\left[(t - t_i^{last}(t)\right]\delta(t - t_j) \; \} \qquad (1.11)$$

$$avec \quad \varepsilon_x = 1 - \exp(-(t_x^{last} - t_x^{last-1})/\tau_x)$$

Les fonctions de suppression sont également représentées par des exponentielles décroissantes où $t^{last}$ et $t^{last-1}$ sont les temps du dernier et de l'avant dernier potentiels d'action générés par un neurone donné. Les cinétiques d'évolution dépendent de la constante de temps $\tau_x$ et peuvent être différentes selon que le potentiel d'action est pré ou postsynaptique. Intuitivement, les fonctions de suppression modulent la force de la synapse selon l'historique des potentiels d'action précédents.

*Saturation de la synapse*

L'équation (1.11) ne borne pas l'évolution du poids synaptique $w_{ij}$. Ce dernier peut croître ou décroître vers des grandes ou petites valeurs arbitraires, et peut même devenir négatif ce qui n'a pas d'équivalent dans la réalité physique. Des facteurs de saturation sont donc ajoutés à l'équation (1.11) pour combler ce défaut (équation 1.12)

$$\frac{dw_{ij}}{dt} = \varepsilon_i \ \varepsilon_j \ \{ \ (w_{LTP} - w_{ij}) \sum_{t_i} P[(t - t_j^{last}(t)] \ \delta(t - t_i) - (w_{ij} - w_{LTD}) \sum_{t_j} Q[(t - t_i^{last}(t)] \ \delta(t - t_j) \ \} \quad (1.12)$$

$$avec \quad w_{LTP} \geq w_{ij} \geq w_{LTD}$$

$W_{LTP}$ et $W_{LTD}$ sont les valeurs de saturation pour la potentiation et la dépression respectivement. Elles sont incorporées dans le modèle de telle sorte que la dépression et la potentiation commencent à s'atténuer lorsqu'elles s'approchent des limites de saturation. Ce qui conduit à une convergence des poids synaptiques vers des valeurs attractives situées entre deux limites. Ce type de saturation est généralement référencé par *soft bound* parce que les poids synaptiques saturent progressivement au court du temps. Par opposition, le terme *hard bound* est utilisé pour référencer les modèles qui laissent évoluer les poids synaptiques arbitrairement jusqu'à arriver aux limites du codage où ils arrêtent brusquement leur évolution.

Ce modèle est le fruit d'une série d'expérimentations biologiques réalisées sur des cellules du néocortex (Destexhe, 2004). Il s'agit d'un modèle biologiquement réaliste lié au contexte expérimental. Les valeurs des paramètres du modèle sont fournies dans le tableau 1.1.

Tableau 1.1. Valeurs des paramètres du modèle de la STDP.

| **Paramètre** | $A_p$ | $A_q$ | $\tau_p$ | $\tau_q$ | $\tau_j$ | $\tau_i$ |
|---|---|---|---|---|---|---|
| **Valeur** | 0.1 | 0.05 | 14.8 ms | 33.8 ms | 28 ms | 88 ms |

## 4 Des outils de modélisation matériels

Le cadre d'étude de cette thèse concerne le développement des outils pour la modélisation neuromorphique qui seront mis par la suite à la disposition des neuroscientifiques. Les systèmes neuromorphiques dédiés à des applications technologiques ne feront pas l'objet de cet état de l'art. La sélection des systèmes neuromorphiques présentés n'est pas exhaustive, mais représentative des approches de conception.

### 4.1 Un simulateur de fonctions cognitives (Schemmel, 2007)

Ce système représente une approche qui exploite les avantages de la technologie CMOS en termes de densité d'intégration, de fréquence d'utilisation élevée et de faible consommation d'énergie. Il s'agit d'un nouveau système neuromorphique à grande échelle faisant des simulations massivement parallèles et hautement accélérées. Il servira d'outil de simulation des grands réseaux de neurones impulsionnels. L'un des buts de ce simulateur et de reproduire des fonctions adaptives du cortex visuel, communément appelé V1.

Ce système implémente le modèle de neurone *Leaky Integrate-and-fire* (I&F) avec des synapses à base de conductance (Destexhe, 1997). Les synapses peuvent être configurées pour changer leurs efficacités selon des règles de plasticité à long et à court terme. Un neurone peut avoir 256 synapses. Les neurones et les synapses correspondantes sont implémentés analogiquement sur un circuit contenant 384 neurones. Plusieurs circuits sont gravés et connectés sur une même plaquette de Silicium. La taille maximale d'un réseau peut atteindre $10^7$ neurones. L'échelle de temps est fonction de la taille du réseau, elle varie entre 1000 et 10000 fois le temps réel biologique.

### 4.2 Winner takes all (Chicca, 2007)

Le *Winner takes all* (WTA) est un principe mis en évidence par des expérimentations biologiques réalisées sur le cortex mammalien (Binzegger, 2004 ; Douglas, 2004). Les réseaux WTA consistent en un groupement de neurones impulsionnels qui entrent en concurrence mutuelle comme réponse à un stimulus. Les neurones possédant la plus forte réponse éliminent les autres et gagnent la compétition. La compétition est réalisée à travers des patterns de connectivité récursifs impliquant des connexions synaptiques inhibitrices et excitatrices. La compétition et la coopération entre les neurones font que l'état de chaque neurone dépend des activités des autres et non plus de sa propre stimulation. Ces réseaux peuvent effectuer non seulement des opérations linéaires, mais aussi non-linéaires (Amari, 1977 ; Dayan, 2001). Les modèles WTA essayent d'émuler les architectures de connectivité corticale et d'étudier leur rôle dans le traitement des informations sensorimotrices et génération des fonctions comportementales.

Les circuits implémentant ces réseaux compétitifs doivent fournir des synapses à dynamique courte et une connectivité récurrente. La puce IFWTA développée par Chicca (Chicca, 2007) incorpore une ligne de 32 neurones de type *Integrate-and-fire*, chacun connecté à une colonne de synapses afférentes. Une colonne contient 14 synapses excitatrices, 2 inhibitrices et 6 locales. Les synapses locales sont utilisées pour la coopération/compétition dans le réseau. La connectivité locale est figée : chaque neurone stimule son premier et second voisin des deux côtés. La puce est équipée par des émetteur-récepteurs permettant de dialoguer avec d'autres circuits selon de protocole AER (Mahowald, 1992). Des applications de sélection et de classification des patterns d'entrée ont été réalisées avec succès.

### 4.3 Un simulateur dynamiquement reconfigurable (Vogelstein, 2007)

L'approche utilisée dans le développement de ce simulateur est de considérer les synapses comme des ressources dynamiques, et non pas comme des ressources dédiées. Dans un circuit neuromorphique typique, chaque neurone est accompagné par une ou plusieurs synapses dédiées, interconnectées entre elles. Cette méthode limite le nombre de cellules et synapses intégrées dans une même puce. Par conséquent, elle est appropriée à l'étude des réseaux neuronaux petits, denses et peu flexibles. L'idée d'utiliser des synapses dynamiquement configurables offre une dimension supplémentaire dans le spectre des réseaux simulables. En effet, gérer l'information synaptique par le biais d'une infrastructure externe, soulage le circuit en effectuant le routage de l'information neuronale et en permettant de manipuler les données synaptiques plus efficacement. Il devient même possible de changer le modèle de la synapse ou de le mettre à jour en cours de la simulation.

Le simulateur consiste en une puce qui incorpore 2400 neurones de type *Integrate-and-Fire*, chacun couplé avec une ″synapse générique″ et un circuit d'émission. Aucune liaison physique n'existe entre les neurones. Les événements émergents transitent sur un bus partagé selon le protocole AER (Mahowald, 1992) et seront routés selon les entrées d'une mémoire programmable de 128 Mo. Chaque entrée de la mémoire contient l'adresse du neurone postsynaptique, la valeur de la conductance et le type de la synapse (excitatrice ou inhibitrice). La valeur de la conductance est calculée par un FPGA selon un modèle synaptique dépendant de l'activité temporelle des neurones. Cette valeur est ensuite transférée à son homologue analogique pour qu'elle soit accueillie par les ″synapses génériques″. Les auteurs de ce simulateur ont profité de la connectivité flexible pour émuler la création et la suppression des synapses pendant l'activité neuronale dans l'hippocampe d'un rat en sommeil.

### 4.4 FPNA : Field-Programmable Neural Array (Farquhar, 2006)

Les circuits programmables, tels que les FPGAs[1] pour les circuits numériques et les FPAAs[2] pour les circuits analogiques, ne cessent d'évoluer en termes de technologie et de miniaturisation. Il devient possible d'intégrer des millions de fonctions élémentaires dans un seul circuit et de les router à volonté. Voici quelques avantages qu'ils fournissent :

- un temps de conception et de fabrication relativement court,
- une robustesse au bruit d'alimentation, la température et les variations technologiques des transistors,
- la configurabilité et la réutilisabilité de ces circuits permettent l'utilisation du même système dans plusieurs applications,
- le circuit peut être optimisé pour chaque réponse à un problème spécifique,
- un interfaçage plus facile avec un ordinateur hôte.

Par analogie, des systèmes identiques sont développés pour la conception des circuits neuromorphiques. Ils offrent une plateforme flexible pour le routage d'un ensemble de circuits neuromorphiques élémentaires. Paul Hasler et ses collègues à *Georgia Tech* ont réalisé une famille de circuits capable d'émuler des modèles de cellules complexes et des petits réseaux de neurones. Ce circuit, appelé *FPNA* pour *Field-Programmable Neural Array*, est une extension d'un FPAA qui contient en plus un ensemble de blocs dédiés au calcul neuromorphique. Son principe d'utilisation est similaire à celui des FPGAs, mais les composants élémentaires sont des circuits intégrés représentant des canaux ioniques, des dendrites et des synapses. Une structure de routage permet de connecter des sorties de cellules (ou des entrées externes) à n'importe quel synapse individuelle dans le circuit, produisant ainsi un dispositif capable d'émuler des réseaux de neurones avec des complexités différentes. Une interface utilisateur facilite le choix du mode d'opération (temps continu ou temps discret) et les spécificités de la conception analogique (mode courant/tension, le rapport signal-à-bruit, la plage dynamique, etc.). Ce genre de circuit permet l'investigation des différents types de calcul individuel dans les cellules nerveuses, et est un exemple de cartographie flexible des réseaux de neurones sur silicium.

### 4.5 Neurogrid : modélisation du cortex (Boahen, 2006)

Pour modéliser le cortex, Boahen et ses collègues de l'université de *Stanford* proposent une grille de circuits contenant 1 million de neurones. Les couches de cellules du cortex, qui sont empilées les unes sur les autres, sont connectées sur 16 circuits identiques carrelés dans une grille, d'où l'appellation *Neurogrid*. Chaque circuit modélise 256x256 cellules à deux compartiments suivant le formalisme de Hodgkin-Huxley, un choix motivé par des similitudes avec le cortex. Des mémoires RAM permettent une connectivité flexible entre les puces (voir figure 1.10), identifient les cibles des événements et spécifient les propriétés anatomiques et physiologiques de chaque connexion (Lin, 2006). Un paramétrage commun des canaux ioniques des cellules d'un même circuit est également possible. Ainsi, *Neurogrid* peut accueillir autant de types cellulaires distincts autant qu'elle contient des circuits.

Les 16 circuits sont connectés selon un arbre binaire par des liaisons de communication fonctionnant à 80 M évènements/s. La communication inter-circuits est gérée par le protocole

[1] Field-Programmable Gate Array
[2] Field-Programmable Analog Array

AER (Boahen, 1998). La topologie de la grille ressemble à celle de la structure en couche observée dans le cortex (voir figure 1.10). Des mémoires RAM incorporées dans chaque circuit configurent les connexions verticales. Une mémoire RAM externe (racine de l'arbre) route les connexions inter-circuits. Les connexions locales (internes au circuit) sont représentées par des projections de fils à travers les cellules de voisinage. Motivés par la localité des cartes d'activation observée dans le cortex, les potentiels postsynaptiques sont programmés par des rayons de voisinage. Par exemple, un rayon de 6 indique que le neurone central peut entrer en interaction avec les 6 neurones les plus proches dans toutes les directions. Neurogrid peut simuler jusqu'à six milliards de synapses.

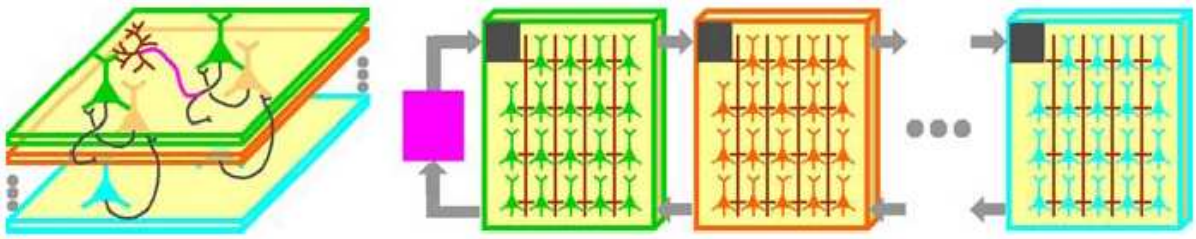


Figure 1.10. Neurogrid, une plateforme pour les simulations corticales. Les modèles de cellules corticales (à gauche) sont implémentés sur des circuits (à droite) avec des neurones analogiques. Des modèles de canaux ioniques émulent des neurones selon le formalisme de Hodgkin-Huxley. Les évènements sont routés par des mémoires RAM à l'intérieur et à l'extérieur des circuits, qui spécifient respectivement les cibles des projections verticales et horizontales (noir et violet), respectivement. Des projections de fils dans les circuits (avec des étendues ajustables) modélisent les connexions locales (marron).

### 4.6 Un simulateur biologiquement réaliste (Renaud, 2010)

Notre équipe de recherche a choisi pour ces systèmes neuromorphiques de satisfaire au mieux le critère de la compatibilité avec la biologie. Cette approche implique l'utilisation des modèles complexes extraits des expérimentations biologiques. L'implémentation des modèles sur des circuits VLSI sur mesure (ASIC[1]) demande plus de ressources et donc limite la taille du réseau. La simulation se fait en temps réel biologique. L'évolution du calcul des modèles de neurones est massivement parallèle. Les paramètres de configuration ont un sens biophysique. Ce type de simulateur présente un lien fort avec les neurosciences. D'un côté, il peut servir comme un outil flexible de modélisation et de simulation réalistes ; d'un autre côté, il peut servir comme un instrument d'interfaçage avec le vivant (Bontorin, 2007). Ceci peut être d'une grande aide pour la surveillance du fonctionnement des tissus vivants puisqu'on garde une maîtrise totale sur la partie artificielle du réseau.

Le modèle de neurone suit le formalisme de *Hodgkin-Huxley*. Les synapses sont à base de modèles à conductance. Chaque neurone possède deux multisynapses (Destexhe, 1999), une excitatrice et une inhibitrice. Les paramètres de configuration d'un neurone règlent les conductances des canaux ioniques et des multisynapses (Buhry, 2008). Le nombre maximal de conductances par neurone est 6. Un neurone peut être connecté à tout autre neurone dans le réseau. Les connexions inter-neurones sont régies par une loi de plasticité. Le modèle de plasticité est de type STDP (Destexhe, 2004) et est implémenté sur des composants électroniques numériques. Il est également configurable et permet de mettre en évidence l'efficacité synaptique à cours et à long terme. Les neurones sont implémentés sur des ASICs analogiques, cinq neurones par ASIC. Plusieurs ASICs sont intégrés sur une même carte

[1] Application Specific Integrated Circuit.

PCB[1]. Plusieurs cartes PCB peuvent être connectées entre elles. La taille du réseau dépend du nombre des cartes connectées et des contraintes de fonctionnement. La taille maximale supportée par le système peut atteindre les 400 neurones.

## 5 Conclusion

Nous avons abordé l'introduction à l'ingénierie neuromorphique par une comparaison entre la simulation numérique classique et la simulation matérielle des réseaux de neurones impulsionnels. Nous avons par la suite souligné l'apport des systèmes neuromorphiques dans les simulations à grande échelle. Après une présentation des notions de base de la neurobiologie, nous avons exposé les modèles du neurone, de la synapse et de la plasticité utilisés dans le cadre de cette thèse.

Une présentation des différentes approches de conception utilisées dans le domaine de l'ingénierie neuromorphique a permis de positionner les travaux ici présentés par rapport à l'état de l'art. Ayant comme priorité principale la fidélité au biologique, l'approche de notre équipe de recherche s'intéresse à la réalisation des simulateurs matériels pour émuler des modèles biologiquement réalistes.

[1] Printed Circuit Board.

# 2. Le simulateur PAX : historique et évolution

Le simulateur de réseaux de neurones impulsionnels développé par notre équipe est nommé PAX, pour *Plasticity Algorithm Computing System*. Il constitue une évolution naturelle des travaux de l'équipe qui a une longue expérience de réalisation des modèles de neurones complexes sur silicium. La première version de ce système date de l'année 2003. Dès lors, plusieurs versions se succèdent marquant des évolutions et améliorations en termes de taille de réseau, réduction des latences et efficacité dans la gestion du réseau. Ce chapitre passe en revue les différentes propriétés de chacune des versions et justifie les choix techniques et technologiques apportés. La section 1 définit le contexte européen du développement de PAX ainsi que l'environnement de réalisation. La section 2 insiste sur les principales caractéristiques et fonctions de base du simulateur. Les différentes versions développées sont décrites en termes de démarche de conception, fonctionnement général, et matériel utilisé dans les sections 3, 4 et 5.

## 1 Le contexte du travail

### 1.1 L'équipe *Ingénierie des Systèmes Neuromorphiques* (ISN)

L'équipe ISN, dans laquelle se sont déroulés les travaux de thèse, s'est récemment scindé en deux équipes : AS2N (*Architecture of Silicon Neural Networks*) et ELIBIO (*ELectronique en Interaction avec la BIOlogie*). La première équipe a pour objectif la conception et la réalisation des circuits et systèmes neuromorphiques proches du biologie. La deuxième s'intéresse à l'interfaçage de l'électronique avec les tissus biologiques. Elle développe des composants électroniques pour l'acquisition et la stimulation de l'activité des tissus neuronaux vivants. Les deux équipes se réunissent sous un objectif commun : la réalisation de systèmes hybrides artificiel-vivant afin d'étudier des mécanismes neuronaux.

*L'équipe AS2N*

Depuis plus d'une décennie, le groupe ISN développe des circuits intégrés à forte composante analogique pour la modélisation en temps réel de l'activité électrique des neurones biologiques. L'équipe AS2N, dirigée par S. Saïghi, prend la suite de ces travaux et continue à produire des circuits neuromorphiques analogiques et mixtes en intégrant des fonctionnalités plus avancées. La génération la plus récente des circuits autorise une configuration plus aisée des conductances des canaux ioniques et permet d'émuler plusieurs types de neurones.

Parallèlement, des recherches sont menées pour caractériser et réaliser une bibliothèque d'IPs[1] analogiques (Saïghi, 2004 ; Levi, 2009). L'objectif est d'optimiser le processus de conception des nouveaux ASICs en privilégiant la réutilisation et en prévoyant les migrations technologiques.

Afin de reproduire le fonctionnement des neurones, une phase de réglage des paramètres des modèles implémentés sur chaque circuit est nécessaire. Des travaux de recherche sur l'estimation et l'optimisation des paramètres analogiques sont menés pour s'accorder avec le modèle biologique (Buhry, 2008).

Dans une seconde étape, les circuits réglés pour simuler des neurones analogiques sont intégrés sur des architectures connexionnistes. L'objectif est de former des réseaux de neurones qui opèrent en temps réel biologique, avec des fonctions de plasticité. Les modèles utilisés sont biologiquement réalistes. Le système global doit être doté d'une haute flexibilité et précision de calcul pour servir d'outil de modélisation pour les neurosciences (Belhadj, 2008a).

Ces travaux sont essentiellement menés en collaboration avec l'unité CNRS de Neurosciences Intégratives et Computationnelles (Institut Albert Fessard, UMR 2191, Gif-sur-Yvette), et le département de la neurophysiologie intégrative (Université *Vrije Universiteit*, Amsterdam). Ces collaborations permettent de couvrir les aspects expérimentaux ainsi que le développement des modèles neurophysiologiques. Une collaboration avec l'équipe *Electronic Vision Group* de l'université de *Heidelberg* (UHEI) est plus orientée vers la modélisation neuromorphique et le développement des standards communs pour les systèmes de simulation. Ces collaborations ont été concrétisées par des projets européens et nationaux, tels que *SenseMaker* (2002-2005), *FACETS* (2005-2009), *FACETS-ITN* (2009-2012), *NeuroversIT* (2006-2009), *PEPS ERNAM* (2008-2009) et PIR NeuroInf ECRéN (2009-2011).

*L'équipe ELIBIO*

L'équipe ELIBIO, dirigée par N. Lewis, s'organise autour de l'interaction des systèmes électroniques avec le vivant. La problématique générale est celle de l'acquisition et du contrôle de l'activité bioélectrique, dans un objectif de compréhension des mécanismes physiologiques ou de contrôle à visée thérapeutique.

Le cadre des applications qui sont développées conduit à proposer une méthodologie de conception des systèmes hybrides vivant/électronique, pour l'acquisition et le contrôle en temps réel de l'activité bioélectrique. Cette problématique nécessite la synergie de plusieurs compétences :

- la maîtrise de l'expérimentation biologique et des caractéristiques physiologiques des tissus, *in vitro* ou *in vivo*,
- la technologie de l'interface vivant/électronique, sa conception et sa modélisation électrique,
- la conception de circuits intégrés analogiques pour le traitement amont du signal bioélectrique ou la génération de stimuli électriques biocompatibles,
- la gestion de la communication en temps réel pour l'ensemble du système bouclé.

Les travaux de l'équipe ELIBIO sont essentiellement menés en collaboration avec l'unité de recherche Mouvement, Adaptation, Cognition (*MAC*) de l'université Bordeaux 2 pour tout

[1] Un IP, ou une propriété intellectuelle, est un terme utilisé dans le domaine de la conception des circuits électroniques pour définir la conception d'une fonction électronique.

ce qui a un rapport avec la neurobiologie, et le groupe de recherche du professeur P. Hasler de l'université *Georgia Tech* (*Atlanta, USA*) pour la réalisation et l'utilisation des circuits de détection et de stimulation. D'autres collaborations à l'échelle régionale et nationale avec des chercheurs d'autres disciplines sont également pérennisées, que ce soit pour l'apport technique ou scientifique. Ces collaborations ont été concrétisées par plusieurs projets ANR, tel que *STN Oscillations* (2009-2013). A l'échelle internationale, l'équipe prêtera attention particulière à l'appel à projets du FP7 Neuro-Bio-ICT attendu pour 2010.

### 1.2 Le projet *SenseMaker*

Planifié sur une durée de 3 ans (2002-2005), le projet *SenseMaker* s'est inscrit dans le cadre de l'appel d'offres *Live-Like Perception Systems* (LPS) et du programme FET (*Future Emerging Technology*), mis en place par la commission européenne. Cet appel d'offres traite du développement de systèmes artificiels sensori-moteurs dont les structures seraient inspirées du vivant.

Le but du projet était de définir et implanter une architecture électronique capable de fusionner des informations sensorielles issues de diverses modalités en une représentation unique dans un environnement donné. L'architecture est directement inspirée des principes de réception et fusion sensorielles du système nerveux. Dans ce projet sont associés des neurophysiologistes, des électroniciens, des informaticiens et des psycho-physiciens de différents instituts de recherche européens.

Dans un premier temps, il s'agissait de dégager les principes de modèles fondamentaux pour l'intégration sensorielle et multi-sensorielles selon une approche biologique et psychophysique. Ces travaux ont été complétés par une approche de modélisation de ces phénomènes d'intégration. Des simulations permettent alors de mettre en œuvre et d'explorer les principes proposés.

L'équipe ISN a intervenu au niveau de l'élaboration d'un système matériel de simulation. La réalisation de ce système constituait les travaux de recherche de doctorat de Y. Bornat et L. Alvado (Bornat, 2006 ; Alvado, 2003). Il s'agissait de la réalisation de la première version du système PAX. Cet outil devrait permettre de proposer et tester des règles optimales de connectivité et de plasticité pour un meilleur traitement des informations sensorielles, en se basant sur des phénomènes observés expérimentalement. La section 3 décrira l'architecture du système.

### 1.3 Le projet *FACETS*

FACETS est un projet quadriennal (2005-2010) européen qui s'inscrit dans la thématique des FET (*Future Emergent Technology*). Il représente une extension du projet *SenseMaker* avec une orientation des objectifs vers la réalisation des nouveaux paradigmes de calcul inspirés des systèmes nerveux biologiques. Le projet réuni 12 équipes de recherche de disciplines différentes. Les équipes peuvent être classées selon 3 catégories de compétences : neurobiologie, modélisation et simulation logicielle, et simulation matérielle. Une partie du projet s'intéresse à une description et simulation précise des différents types de cellules nerveuses (échelle cellulaire), tandis qu'une deuxième partie s'intéresse aux dynamiques de réseaux de neurones à petite et à grande échelle (échelle du réseau). La figure 2.1 illustre cette organisation et les interactions entre les différentes parties.

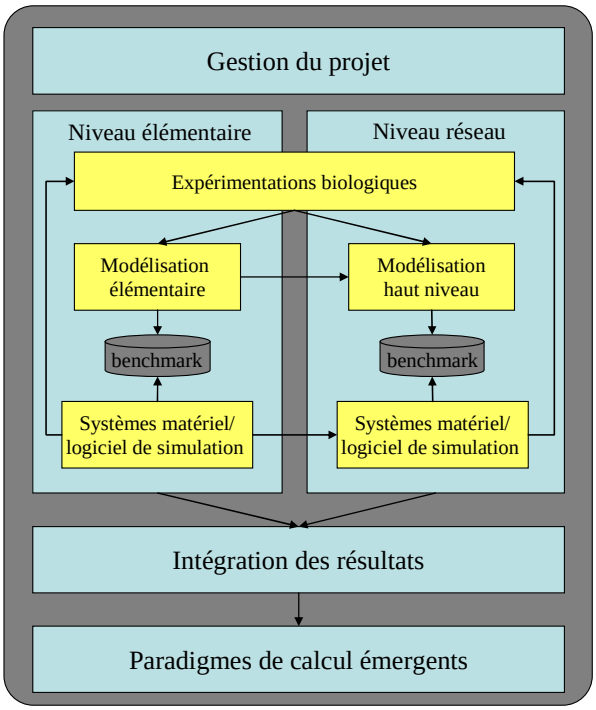



Figure 2.1. Organisation au sein du projet FACETS.

La recherche dans le projet FACETS suit plusieurs approches :

- Des expérimentations *in-vitro* sur des cultures de neurones : les données et les résultats sont collectés dans une base de données, et sont accessibles pour tous les membres du projet,
- Analyses statistiques et mise en équations des données expérimentales : des modèles mathématiques sont extraits à partir des données expérimentales enregistrées,
- La simulation logicielle de ces modèles,
- La conception, l'implémentation et l'utilisation des systèmes neuromorphiques qui implémentent physiquement les modèles neuronaux.

Les approches sont interdépendantes horizontalement et verticalement (voir figure 2.1). Les résultats finaux sont un éventail de découvertes biologiques expérimentales, des modèles à l'échelle de la cellule et du réseau, et des systèmes neuromorphiques. Ces résultats permettent non seulement l'investigation de nouveaux paradigmes de calcul neuronal, mais aussi d'alimenter en retour les expérimentations biologiques.

L'équipe ISN se situe dans la troisième approche du projet, accompagné par l'équipe de l'université de *Heidelberg*. Deux orientations de conceptions différentes émergent et créent la diversité de la conception et la réalisation des systèmes neuromorphiques au sein du projet. L'équipe ISN réalise des circuits analogiques pour modéliser des neurones avec un haut degré de précision. Cette équipe s'intéresse aux réseaux de petite taille (quelques centaines de neurones au maximum) pour exploiter le réalisme des modèles de calcul neuronal et la flexibilité de la configuration des paramètres des modèles. Le système proposé fonctionne en temps réel biologique, ce qui offre une opportunité pour établir des systèmes hybrides, i.e. un réseau constitué de neurones neuromorphiques et biologiques qui communiquent ensemble (Bontorin, 2007). D'un autre côté, l'équipe de *Heidelberg* essaye

d'exploiter les possibilités offertes par la technologie CMOS en termes de densité d'intégration d'un circuit VLSI, d'une fréquence d'utilisation élevée et d'une faible consommation d'énergie pour réaliser des simulations neuronales à grande échelle et massivement accélérées en comparaison avec l'échelle de temps biologique (Schemmel, 2007). Il est attendu que leur système aidera à combler des limitations de simulation logicielle en terme de rapidité et taille des réseaux.

## 2 Le simulateur PAX (Plasticity Algorithm Computing System)

La modélisation et la simulation sont des outils pour reproduire des faits réels complexes. L'efficacité d'une simulation réside dans son degré de réalisme par rapport au fait réel. La pertinence d'un simulateur neuromorphique se mesure par son habilité à reproduire les mesures observées en biologie. Cependant, vue la complexité des systèmes biologiques, on se trouve obligé de négliger certains détails. Il en résulte que l'on perd en termes de précision dans le fonctionnement et de compatibilité avec la structure originale.

L'abstraction est une approche naturelle pour éliminer des détails jugés non utiles dans la reproduction d'un phénomène donné. Cependant, toute abstraction doit être justifiée par la pertinence des résultats finaux et la simplicité qu'elle apporte. Le degré de la compatibilité et de précision d'un système neuromorphique dépend des niveaux d'abstraction utilisés et de leur concordance avec les faits réels.

### 2.1 Les abstractions dans PAX

On peut classer les abstractions selon quatre types : abstraction fonctionnelle, temporelle, de données ou structurelle. Ci-dessous une brève description de chaque type d'abstraction suivie par une liste non-exhaustive des abstractions utilisées dans PAX.

*Abstraction fonctionnelle*

L'abstraction fonctionnelle permet une spécification partielle du comportement réel et autorise l'implantation partielle de certaines fonctions. Pour modéliser un système biologique, deux grands niveaux d'abstraction fonctionnelle apparaissent : le niveau physiologique et le niveau phénoménologique. Le niveau physiologique comporte les détails sur les mécanismes physiologiques qui régissent un comportement donné. Ce niveau peut être subdivisé en plusieurs sous-niveaux, permettant l'insertion de plus de détails dans les niveaux les plus bas. Un exemple typique est la modélisation du mécanisme de génération d'un potentiel d'action d'un neurone par l'intermédiaire des canaux ioniques de la membrane cellulaire. Une première abstraction peut se faire en représentant tous les canaux ioniques de même type par une conductance ionique unique. Ainsi, on aura besoin d'une conductance pour représenter des millions de canaux ioniques de même type. Une deuxième abstraction peut se faire lorsqu'on suppose que les canaux ioniques sont localisés dans un seul emplacement de la membrane cellulaire (modèle mono-compartimental) et ne sont pas distribués sur son contour (modèle multi-compartimental). Les modèles définis à ces niveaux d'abstraction sont appelés modèles physiologiques.

Le niveau phénoménologique est plus abstrait que le niveau physiologique. Le mécanisme derrière la production d'un phénomène biologique n'est pas pris en compte, seul le comportement externe est retenu. Les modèles qui en découlent sont dit modèles phénoménologiques et sont généralement des interprétations mathématiques des résultats empiriques. L'exemple de la génération d'un potentiel d'action par un neurone se réduit à une capacité qui, à un certain seuil de tension, se décharge. La forme du potentiel d'action n'est

pas représentée, seule sa présence ou non est indiquée. Ce type d'abstraction peut être utile pour réaliser des simulations à grande échelle.

*Abstraction temporelle*

L'abstraction temporelle consiste à observer les comportements qui dépendent du temps avec une certaine granularité. La finesse ou la grosseur de cette granularité dépend du niveau d'abstraction. Ainsi, à un haut niveau d'abstraction on peut négliger les délais de propagation des potentiels d'action au long des axones et dendrites, alors que dans un niveau plus bas, on tient compte du moindre délai. Cette abstraction permet de déterminer la rigueur de l'aspect temps-réel dans un système neuromorphique.

*Abstraction des données*

Cette abstraction favorise une vue abstraite des données utilisées. Elle peut représenter les réactions chimiques ou physiques dans un système biologique. Ainsi, la dynamique de la membrane d'un neurone peut être résumée en une entité booléenne, '0' si le potentiel de membrane ne dépasse pas un certain seuil, '1' dans le cas contraire. D'une autre part, l'efficacité de la transmission et la réception des neurotransmetteurs au niveau de la jonction synaptique peut être représentée par un poids synaptique. Le degré de précision de la représentation du poids définit le niveau de l'abstraction des données de l'efficacité synaptique.

*Abstraction structurelle*

Ce mécanisme d'abstraction supprime les détails sur la structure effective du système et favorise une vue globale dans le but de maîtriser sa complexité. Un exemple typique de l'abstraction structurelle est l'utilisation d'une structure mono-compartimentale d'une cellule au lieu de sa structure multi-compartimentale biologique. Tous les détails sur la structure volumique de la cellule sont supprimés et remplacés par un point dans l'espace. Cette abstraction est largement utilisée dans la simulation matérielle des réseaux de neurones impulsionnels (Renaud, 2007 ; Schemmel, 2007 ; Indiveri, 2007). Un autre exemple d'abstraction structurelle est le remplacement de la grande densité des connexions point-à-point entre les neurones d'un tissu vivant par quelques connexions partagées. Ceci est souhaitable dans la conception des systèmes neuromorphiques puisque la différence est grande entre la fréquence de génération des potentiels d'action dans le milieu biologique (quelques Hz ou dizaines de Hz) et la fréquence de fonctionnement des bus de communication électronique (quelques dizaines de MHz) ; ce qui permettra de multiplexer plusieurs connexions point-à-point dans le temps en utilisant un seul support de communication physique.

*Un calcul à des niveaux d'abstraction différents*

Les systèmes biologiques sont organisés hiérarchiquement d'une façon complexe. La compréhension des mécanismes qui les régissent ne peut se faire sans avoir recours à une mise en modèle abstraite de leur fonctionnement réel. Le niveau d'abstraction utilisé dépend de la complexité du mécanisme étudié et de son rôle dans l'application cible. Ceci veut dire qu'on est amené à utiliser des modèles à différents niveaux d'abstraction pour représenter un système biologique complet.

D'une autre part, il faut montrer que la combinaison des modèles utilisés donne des résultats efficaces en termes de calcul et reproductivité des mécanismes et phénomènes biologiques étudiés. Dans certaines situations, la combinaison d'un modèle complexe, avec un deuxième plus simple, masque la précision du premier, et le degré de réalisme retenu sera

celui du plus simple. Le défi sera de rechercher un équilibre fonctionnel entre les niveaux d'abstraction hétérogènes des modèles utilisés.

Les ressources matérielles induisent une limitation immédiate de l'implémentation des modèles. L'exploration architecturale et la répartition des ressources doivent donc se faire en même temps que la définition des modèles. Une bonne répartition permet d'éviter les goulets d'étranglement dans le système global et d'assurer une meilleure fluidité des données.

*Les abstractions dans PAX*

Le simulateur PAX vise à réaliser une simulation matérielle en mettant l'accent sur les phénomènes d'adaptation et d'apprentissage, les dynamiques de la production des potentiels d'action, la stimulation synaptique et l'interaction entre les neurones. L'objectif est de reproduire ces fonctions avec un degré de réalisme élevé. Pour cette raison, les abstractions appliquées sont principalement utilisées pour simplifier l'implémentation matérielle des modèles et non pas les modèles eux-mêmes.

Au niveau cellulaire, le soma suit le formalisme de Hodgkin-Huxley (Hodgkin et Huxley, 1952). Une première abstraction consiste à négliger les effets d'atténuation des potentiels neuronaux et les propriétés intrinsèques de la membrane : la membrane est considérée comme un seul compartiment. Cette disposition suppose implicitement que les canaux ioniques de même type sont regroupés dans une même conductance. Ceci simplifie l'implémentation matérielle d'un neurone par rapport aux cas d'une modélisation multi-compartimentale. Une deuxième abstraction consiste en une amplification des valeurs électriques des paramètres biologiques pour rendre les valeurs de courant manipulables par les technologies des circuits intégrés et améliorer la dynamique du système.

Cependant, il n'y a pas d'abstraction temporelle à ce niveau. L'évolution des circuits en temps continu et en temps réel biologique est similaire à celle de leurs homologues biologiques. Ce choix ne pose pas de problème de réalisation matérielle et permet un interfaçage avec des cellules vivantes.

Au niveau réseau, plusieurs abstractions sont utilisées. Premièrement, on a procédé à une abstraction fonctionnelle du comportement physiologique d'une synapse. La physiologie des mécanismes qui se produisent au niveau de la jonction synaptique sont très complexes. On se limitera à représenter les phénomènes résultants de ces mécanismes à l'aide d'un modèle phénoménologique réaliste (Destexhe, 2004). La plasticité synaptique agit par conséquent sur une grandeur arithmétique, le poids synaptique.

Une deuxième abstraction est appliquée à la synapse. Il s'agit d'une abstraction structurelle qui permet d'éviter la grande densité des liaisons synaptiques. Les entrées synaptiques pour un même neurone sont regroupées dans une synapse ″collective″, appelé multisynapse (Destexhe, 1999). La multisynapse somme toutes les stimulations causées par les neurones présynaptiques de même nature (inhibiteurs ou excitateurs) pour stimuler la membrane cellulaire du neurone cible.

Une troisième abstraction concerne la structure du réseau. Les connexions synaptiques qui relient les neurones sont attribuées d'une façon virtuelle : il n'y a pas de liaisons physiques réelles entre les cellules. Chaque neurone est doublement connecté, en entrée et en sortie, à un commutateur. Ce dernier récupère les évènements des sorties des neurones, et transmet les stimulations synaptiques vers les entrées des neurones cibles selon la connectivité du réseau. La connectivité détermine quel neurone est connecté à quel(s) autre(s) neurone(s). C'est l'utilisateur final qui fixe la connectivité du réseau à l'aide de la *matrice de connectivité*. Le commutateur utilise le multiplexage temporel pour gérer les flux de données. Cette technique est connue sous le nom de *synapse virtuelle*.

Une dernière abstraction que l'on citera ici concerne les potentiels d'actions. Etant donné que la forme des potentiels d'action apporte peu, on attribuera des valeurs booléennes pour indiquer la présence ou pas d'un potentiel d'action. Ceci facilitera le traitement et la transmission des évènements des neurones.

Enfin, pour des raisons de compatibilité avec le niveau cellulaire, il n'y a pas d'abstraction temporelle apportée au niveau réseau. On garde toujours une opérabilité en temps réel biologique. Ceci permet de garder un équilibre entre le modèle de la cellule et les modèles niveau réseau. La section suivante montre avec plus de détails les interactions entre les deux niveaux.

### 2.2 La démarche méthodologique

Toutes les versions de PAX suivent une démarche commune. Seule la réalisation diffère. La démarche concerne la répartition du calcul neuronal, le principe de fonctionnement et l'interfaçage des différentes parties du système.

*Cœur analogique pour le calcul neuronal*

Le cœur de calcul neuronal doit satisfaire des contraintes de compatibilité avec des modèles biophysiques qui correspondent aux populations majoritaires de neurones peuplant le néocortex. Les phénomènes dépendants du temps doivent se faire en temps réel biologique. Les grandeurs électriques sont proportionnelles aux grandeurs biologiques. Le paramétrage des composants de calcul doit être flexible et facilement modifiable permettant la configuration de plusieurs catégories de neurones (Bornat, 2006 ; Saïghi, 2004).

L'évolution dans les circuits analogiques s'adapte bien avec les contraintes de fonctionnement souhaitées pour le calcul neuronal et offrent une représentation fidèle au biologique. Les cœurs de calcul seront donc des composants analogiques configurables.

*Gestion numérique du réseau*

La principale caractéristique recherchée pour le système est une entière reconfigurabilité du réseau. Cela signifie que toutes les possibilités de connexion inter-neurones doivent être couvertes, en allant d'un réseau non-connecté à un réseau complètement connecté (ou réseau *all-to-all*). Il est aussi souhaitable d'avoir la possibilité d'ajouter des informations concernant chaque connexion : plastique ou non, excitatrice ou inhibitrice, poids synaptique initial, etc.

Les connexions seront régies par un algorithme de plasticité. Les modèles de plasticité sont en changement perpétuel. Il n'existe pas de modèle générique qui englobe tous les mécanismes de plasticité observés dans la biologie. A chaque découverte d'un nouveau mécanisme plastique, un nouveau modèle pourrait controverser l'ancien. Il devient donc nécessaire d'éviter une implémentation figée de l'algorithme de plasticité. Une architecture reprogrammable permettra une mise en œuvre rapide des changements éventuels apportés sur le modèle de plasticité. En ajoutant le fait que, quelle que soit sa nature, le système de simulation sera géré par une interface numérique, il devient évident d'utiliser le domaine numérique pour la gestion du réseau.

*Interfaçage numérique-analogique*

La mise en réseau de plusieurs cœurs de calcul nécessite un interfaçage entre leurs composants analogiques et la gestion numérique du réseau. Le potentiel de membrane passe par un comparateur qui le transforme en une impulsion numérique. Cette impulsion passe à 1 si le potentiel de membrane dépasse un certain seuil. Il en résulte que la largeur de l'impulsion code la durée d'un potentiel d'action. Cette impulsion marque l'apparition d'un potentiel d'action dans le domaine numérique.

Dans le sens inverse, il s'agit de stimuler le potentiel de membrane par un courant synaptique. Ce dernier dépend de l'efficacité synaptique de chaque connexion. Cette efficacité est calculée numériquement par l'algorithme de plasticité puis est transformée en une impulsion numérique dont la largeur code la valeur du poids synaptique. Un courant, appelé courant synaptique, augmente son intensité tant que l'impulsion reste au niveau logique '1'. Le caractère inhibiteur ou excitateur du courant synaptique est déterminé par son signe, négatif ou positif. Une fois que l'impulsion revient au niveau logique '0', le courant synaptique décroît selon le modèle de la multisynapse présenté dans la section 3 du chapitre 1. La figure 2.2 résume l'architecture générale du simulateur PAX.

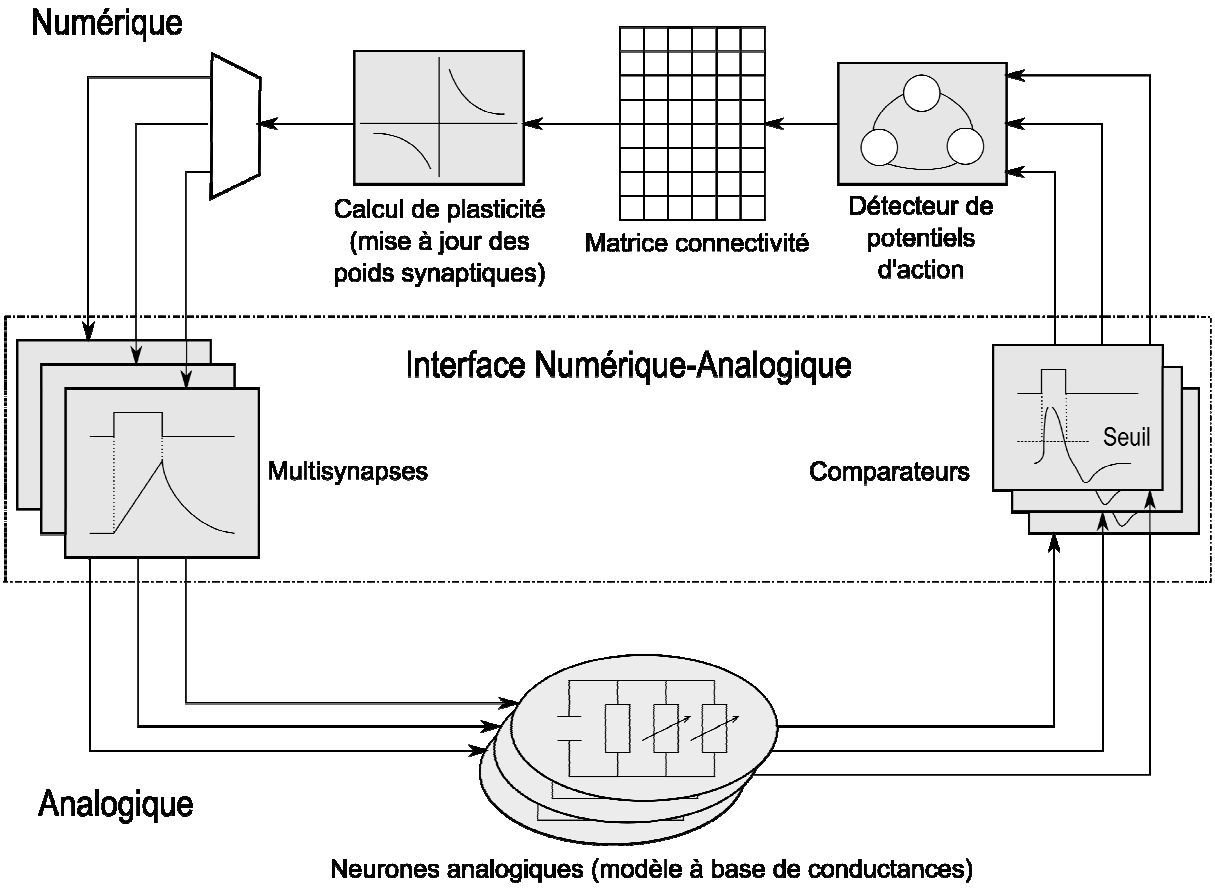



Figure 2.2. Schéma général de PAX : les neurones sont intégrés sur silicium, tandis que la gestion du réseau (connectivité, plasticité) s'effectue numériquement. Une interface numérique-analogique assure la communication entre les deux représentations.

### 2.3 Les fonctions neuronales

Le simulateur PAX contient des fonctions de base nécessaires pour la réalisation des objectifs prescrits dans le cahier des charges. Ces fonctions constituent le noyau de calcul principal du système. Il est tout de même possible d'incrémenter le noyau par des fonctions neuronales supplémentaires afin de satisfaire des contraintes d'opération plus avancées ou de réaliser des simulations plus proches de la biologie.

Les fonctions de bases de PAX sont :

- Les fonctions de plasticité et de connectivité du réseau,

- Les fonctions de calcul des dynamiques de génération des potentiels d'action à base des modèles à conductances,
- Les fonctions de stimulation et d'intégration synaptique.

Des exemples de fonctions supplémentaires peuvent être :

- Le bruit synaptique : pour reproduire avec plus de précision le comportement d'un neurone ou d'un réseau de neurones tel qu'il est dans son milieu naturel, il est nécessaire de prendre en compte l'influence de l'activité électrique des neurones du voisinage.
- Le délai axonal : la propagation de l'information neuronale tout au long d'un axone dépend, entre autres, de sa longueur et sa conductivité. Le délai axonal permet d'attribuer des valeurs à ces paramètres. Une dimension spatiale est donc ajoutée au réseau de neurones simulé.
- La connectivité dynamique : dans les réseaux de neurones biologiques les connexions ne sont pas figées, elles se prolongent ou se rétrécissent constamment dans le substrat. Il en résulte que des nouvelles connexions se créent, alors que d'autres disparaissent. La connectivité dynamique d'un système neuromorphique permet de changer la topologie d'un réseau durant l'exécution de la simulation.

L'ajout des fonctions supplémentaires dans PAX dépend des ressources de calcul disponibles et des contraintes de simulation. En effet, certaines fonctions sont très couteuses en termes de mémoires et en temps de traitement. A titre d'exemple, pour associer un délai axonal à chaque connexion dans un réseau de N neurones, il faudra stocker dans la mémoire $N^2$ valeurs représentants les délais de $N^2$ connexions. A chaque fois qu'un potentiel d'action présynaptique est généré, il y a une possibilité qu'il entre en interaction avec N neurones postsynaptiques (réseau complètement connecté), et donc utilise N valeurs de délais pour déterminer le temps d'arrivée du potentiel d'action aux neurones postsynaptiques. Supposons que chaque neurone provoque cette situation à une fréquence de 100 Hz. On aura dans ce cas $100*N^2$ délais à traiter par seconde. Le traitement d'un seul délai axonal nécessite son stockage dans une mémoire et une opération de comparaison permanente pour décider du moment de l'envoi du potentiel d'action vers le neurone cible. Le traitement en parallèle de cette fonction est quasiment impossible pour des réseaux de grande taille à cause du nombre important des blocs de logique combinatoire nécessaires pour la réalisation. L'approche séquentielle permet de réduire le nombre d'opérations logiques utilisées mais elle risque de causer de grands délais de calcul à cause de l'accumulation des potentiels d'actions en attente de traitement.

### 2.4 Les contraintes de fonctionnement

L'établissement d'un outil de simulation matérielle utile pour les neurosciences nécessite :

- Une preuve de sa pertinence biologique : il doit être capable de générer des structures et des comportements biologiquement réalistes selon le niveau de détail et de précision avec lesquels il est conçu.
- Une flexibilité dans le jeu de paramètres : un système hautement configurable qui permet de simuler des situations difficilement mesurables par les expérimentations biologiques.
- Une opérabilité facile par les non-experts en électronique.

Pour satisfaire ces critères, chaque simulateur neuromorphique est soumis à des contraintes de fonctionnement qui garantissent les résultats souhaités. A travers le simulateur

PAX, on propose de reproduire les phénomènes observés en biologie à l'échelle cellulaire et à l'échelle du petit réseau. Nous avons en conséquence fixé les contraintes de fonctionnement suivantes.

*Temps réel biologique*

Comme mentionné précédemment, le temps réel biologique doit être respecté dans toutes les parties fonctionnelles du système PAX. L'échelle temporelle de l'activité électrique dans les systèmes biologiques doit être similaire à celle de l'activité électrique dans les circuits et systèmes fabriqués.

On sera amené dans les versions du système PAX à reproduire l'évolution temporelle de la formation d'un potentiel d'action, de la plasticité synaptique et du délai axonal. Les résolutions temporelles correspondantes observées en biologie sont résumées dans le tableau ci-dessous.

Tableau 2.1. Les échelles temporelles des fonctions neuronales en biologie.

| | **Résolution temporelle** |
|---|---|
| **Durée d'un potentiel d'action** | 1 ms – 5 ms (Tritsch, 1998) |
| **Mécanismes synaptiques** | 10 µs – 10 ms (Gerstner, 2002) |
| **Délai Axonal** | 10 µs – 150 ms (Tritsch, 1998) |

*Précision de calcul*

Le degré de précision requis pour le calcul d'une fonction neuronale dépend du niveau d'abstraction dans lequel elle est spécifiée. A titre d'exemple, si en fait une erreur de quelques millisecondes pour polariser et dépolariser le potentiel de membrane d'un neurone fonctionnant en temps réel biologique, la forme du potentiel d'action généré ne sera pas correcte. Par contre, si on fait une erreur de quelques microsecondes, l'effet de l'erreur influera peu ou pas la forme finale du potentiel d'action. On dit que le calcul est ″suffisamment″ précis.

En règle générale, on peut dire qu'un calcul neuronal est suffisamment précis si les conditions suivantes sont vérifiées :

- L'erreur de calcul n'influe pas le résultat final de la fonction et préserve l'intégrité du mécanisme neuronal simulé,
- Le calcul effectué est capable de représenter les variations clés dans les grandeurs simulées susceptibles de provoquer un changement d'état de la fonction.

Les erreurs de calcul à effet boule de neige (erreur qui croît en se propageant) ne sont pas détectées par ces conditions. En effet, une erreur de calcul qui vérifie les conditions ci-dessus peut s'amplifier en cours de simulation et causer des comportements incohérents. Il est donc nécessaire de contourner la résolution de ce type d'erreurs autrement qu'en augmentant la précision de calcul.

*Configuration flexible*

L'un des points clés du système PAX est la grande flexibilité apportée par la souplesse de configuration de ses composants. La contrainte de configurabilité spécifie que toute implémentation d'une fonction neuronale doit être capable de représenter tous les comportements décrits par le modèle par simple changement du jeu des paramètres. Ainsi, des neurones décrits selon le modèle de Hodgkin-Huxley doivent être capable de représenter les types de neurones que couvre ce modèle (Izhikevich, 2004).

## 3 PAX1 : une plasticité logicielle

Ce simulateur a été conçu dans le cadre des travaux de thèse de Y. Bornat au sein de l'équipe ISN (Bornat, 2006). Bien que n'ayant pas été réalisé par l'auteur, ce système est tout de même présenté pour illustrer l'évolution du simulateur PAX.

### 3.1 Vue d'ensemble

Cette version satisfait les contraintes temps réel, précision de calcul et souplesse de configuration. Elle supporte deux fonctions neuronales supplémentaires : le bruit synaptique et le délai axonal. Cependant, le nombre de neurones maximal qu'elle peut gérer ne dépasse pas la dizaine.

Le système peut être décomposé en 3 couches : une couche logicielle et deux couches matérielles. Les couches matérielles se divisent en couche analogique et couche numérique. Cette dernière sert de relais entre le programme logiciel et les circuits analogiques. Elle pilote, d'une part, les périphériques de liaison avec la machine hôte, et d'autre part, elle assure l'interfaçage analogique-numérique (voir figure 2.3).

La boucle de simulation représente les chemins tracés par l'information neuronale lors de la simulation. Dans le système PAX1, les trois couches sont impliquées dans la boucle de simulation. La couche logicielle traite la connectivité et la plasticité du réseau, ainsi que la génération du bruit synaptique. Elle reçoit, à partir de la couche matérielle numérique, les numéros des neurones qui ont généré des potentiels d'action, accompagnés de leur heure d'apparition (*timestamp*). Un programme logiciel vérifie l'existence des connexions synaptiques relatives aux numéros de neurones reçus et réalise un calcul des règles de plasticité. A la fin du calcul, les nouvelles valeurs des poids synaptiques sont envoyées vers la couche matérielle numérique. Cette dernière transforme les poids synaptiques en des impulsions de largeur variable pour stimuler les multisynapses des circuits analogiques localisés dans la couche matérielle analogique. Le courant de stimulation fait varier le potentiel de membrane et déclenche éventuellement l'apparition d'un potentiel d'action. Ce dernier sera détecté par la couche numérique et envoyé, après expiration de son délai axonal, vers le logiciel de traitement de plasticité. Le même cycle se répète jusqu'à la fin de la simulation.

Le passage de la couche matérielle numérique à la couche logicielle, dans un sens ou dans l'autre, constitue le premier goulet d'étranglement de cette version de PAX. Malgré le fait que les liaisons standards des machines hôtes soient sophistiquées et fonctionnent à haut débit, elles incluent des latences relativement importantes. Un deuxième goulet d'étranglement réside dans le calcul logiciel. Il est en effet difficile de basculer d'un calcul parallèle vers un calcul séquentiel sans perdre en termes de temps de traitement. L'enjeu principal sera donc de réduire le coût temporel du traitement des données. Des systèmes d'exploitation temps-réel ont été utilisés pour réduire les latences causées par les couches logicielles de la machine hôte (Bornat, 2006).

Le grand avantage de cette version réside dans la flexibilité et la souplesse de l'implémentation de l'algorithme de plasticité et la gestion de la connectivité qu'apporte la réalisation en logiciel. Il est donc possible d'étudier plusieurs types d'apprentissage et d'adaptation dans les réseaux de neurones biologiques.

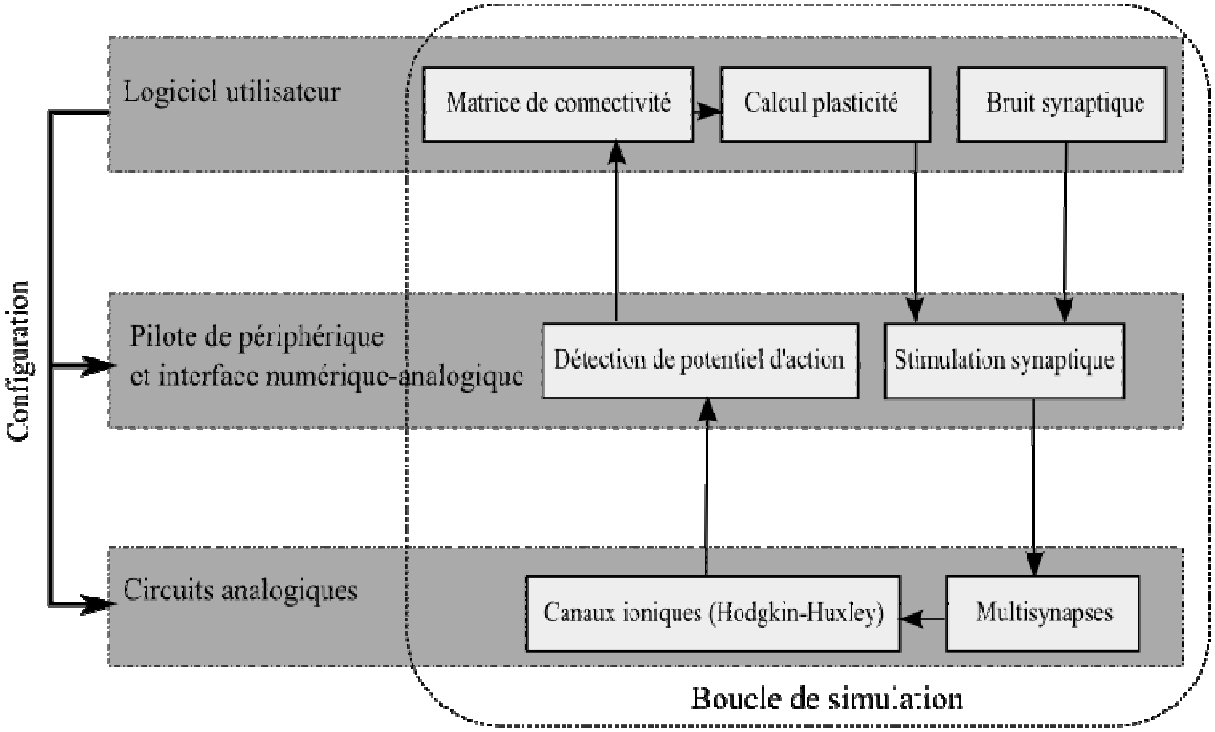



Figure 2.3. Répartition de calcul dans les couches de PAX1

### 3.2 La carte PAX1

Tout le système matériel est réalisé sur une même carte, nommée carte PAX1. La figure 2.4 montre le schéma de principe de la carte. La couche matérielle analogique est représentée par des circuits intégrés nommés *Trieste*[1]. La couche matérielle numérique est représentée par un FPGA de marque *Xilinx Spartan IIE*. Le FPGA est connecté, d'un coté, aux 8 circuits analogiques et, d'un autre coté, à un bus PCI[2]. Le bus PCI sert à connecter la carte PAX à la machine hôte dont le choix du processeur s'est porté sur le *Pentium 4*. Le pont dédié à la connexion au bus PCI est assuré par le composant *PCI9056* de la société *PLX*. Tout le système est cadencé avec une horloge de 64 MHz. La figure 2.5 fournit une photographie de la carte PAX1.

Vu le nombre modeste des circuits analogiques embarqués, ces derniers sont directement câblés avec le FPGA grâce à leurs entrées/sorties numériques. En tant que relais, le FPGA stocke et met en forme les données pour les rendre disponibles sur la carte. Il peut s'agir de convertir des données parallèles en série pour les convertisseurs numérique/analogique, de multiplexage avec des paramètres de circuits ou encore de la conversion d'un entier en largeur d'impulsion, pour traduire les valeurs des poids synaptiques avant de les envoyer sur les entrées synaptiques des circuits.

[1] Trieste est le nom d'une ville en Italie. Le droit d'attribution des noms des circuits est attribué au concepteur (L. Alvado).
[2] Peripheral Component Interconnect

Pour mieux gérer la quantité de données qui transitent sur le système, une mémoire SRAM externe est connectée au FPGA. Elle servira essentiellement pour le stockage des données des applications supplémentaires (bruit synaptique, délai axonal) et des données de gestion du réseau (historique). Des connecteurs regroupant 60 entrées/sorties sont également connectés au FPGA pour une éventuelle utilisation ou extension du système.

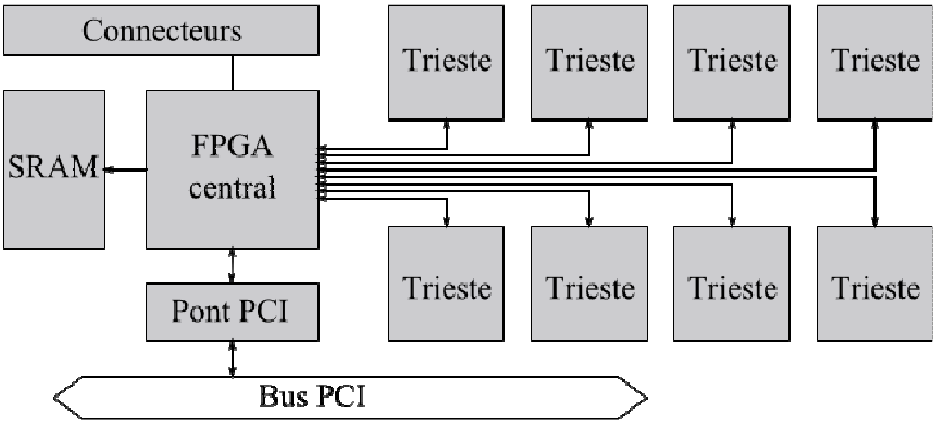



Figure 2.4. Schéma de principe de la carte PAX1

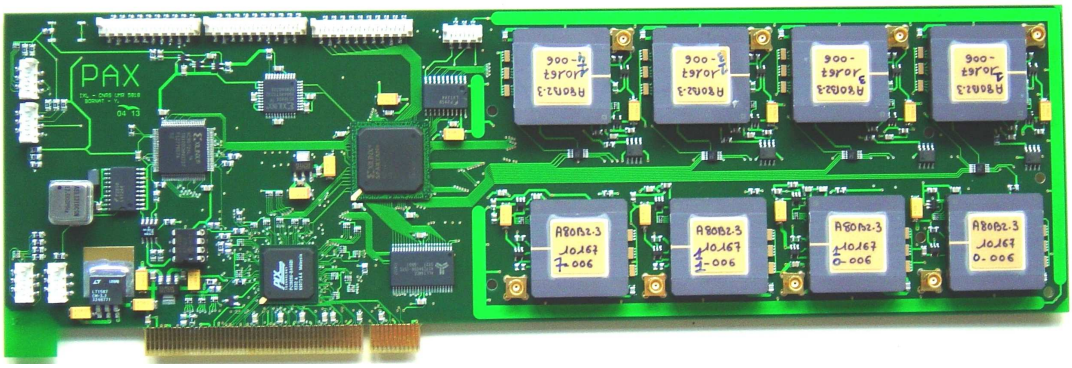



Figure 2.5. Photographie de la carte PAX1

### 3.3 Le circuit *Trieste (Alvado, 2003)*

Le circuit *Trieste* est l'un des premiers circuits développés par l'équipe ISN intègrant les outils numériques nécessaires pour l'intégration en réseau. Il incorpore un seul neurone et deux synapses. Sa fabrication a été effectuée en technologie BiCMOS 0,8µm chez *austria*micro*systems* et il utilise une surface de silicium de 11,5mm$^2$.

Ce circuit utilise le formalisme de Hodgkin-Huxley comme modèle étendu du neurone. Il est capable d'émuler deux types de neurones : les neurones excitateurs (également qualifiés de *RS* pour *Regular Spiking*) et les neurones inhibiteurs (ou *FS* pour *Fast Spiking*) qui peuvent reproduire le phénomène d'adaptation grâce à un canal modulant.

Pour mieux s'adapter à la technologie électrique utilisée (BiCMOS), l'échelle des grandeurs biologiques a été adaptée pour un meilleur fonctionnement du circuit. Les tensions sont multipliées par 10, les courants par 100, les conductances par 10 et les capacités par 10. Pour simuler la polarisation et la dépolarisation de la membrane cellulaire, le potentiel de repos (0 V biologique) est fixé à 1,8 V électrique.

L'architecture du circuit est présentée dans la figure 2.6. Le cœur de simulation est une suite de conductances représentant les canaux ioniques. Dans le souci de s'affranchir d'une mise en œuvre complexe, les paramètres des conductances sont figés. Un certain nombre de paramètres numériques sont, cependant, contrôlables par l'utilisateur final. Ils permettent de sélectionner ou désélectionner les conductances d'un neurone afin de changer son type. Les entrées synaptiques sont connectées directement au cœur de simulation. On trouve deux multisynapses pour chaque neurone, une excitatrice et une inhibitrice, qui correspondent aux deux types de neurones présents dans les réseaux construits avec ce circuit. Et enfin, pour la visualisation de l'évolution des canaux ioniques et du potentiel de membrane, des sorties analogiques peuvent être directement connectées à un oscilloscope.

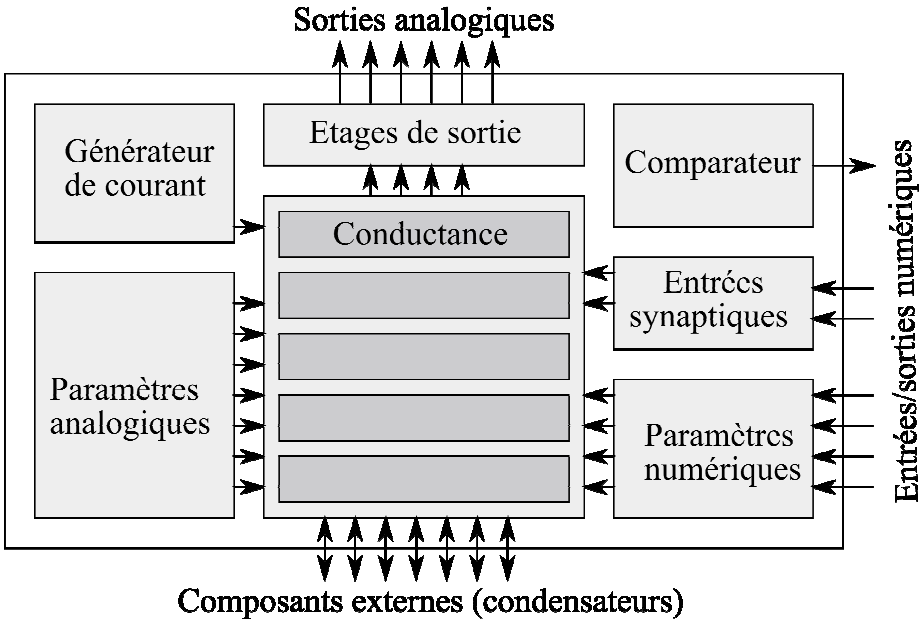



Figure 2.6. Schéma bloc du circuit *Trieste*

## 4 PAX2 : le système purement matériel

Cette version de PAX a été commencée par Y. Bornat dans le cadre de sa thèse et achevée par l'auteur. Dans cette version, on propose d'intégrer un plus grand nombre de neurones tout en satisfaisant les mêmes contraintes temporelles et de précision. La démarche était de migrer vers un calcul purement matériel en vue de réduire les latences de fonctionnement dû à l'intervention du logiciel.

### 4.1 Vue d'ensemble

Cette version satisfait les contraintes temps réel, précision de calcul et flexibilité de configuration. Elle supporte une fonction neuronale supplémentaire : le bruit synaptique. Elle a principalement servi comme un prototype pour une version ultérieure. Le système a été utilisée par A. Daouzli dans ses travaux de thèse pour étudier des mécanismes de plasticité appliqués à des séquences de potentiels d'action corrélés (Daouzli, 2009).

De même que PAX1, ce système peut être décomposé en 3 couches : une couche logicielle et deux couches matérielles (analogique et numérique). La couche logicielle n'intervient plus dans la boucle de simulation. Elle sert uniquement à générer et envoyer les paramètres de configuration, ainsi que les séquences du bruit synaptique. A la différence du système PAX1, le calcul de la plasticité et la gestion de la connectivité s'ajoutent à la gestion des

périphériques et l'interfaçage analogique-numérique dans la couche numérique matérielle. La figure 2.7 résume la répartition des tâches de calcul dans le système PAX2.

La boucle de simulation est réduite à des transitions entre les couches matérielles. A l'arrivée d'un potentiel d'action, la couche numérique vérifie les connexions synaptiques relatives au neurone actif et calcule leur poids synaptiques. Les valeurs de ces derniers seront transformées en des impulsions pour activer les entrées synaptiques des circuits analogiques. L'activation des synapses peut causer la génération des nouveaux potentiels d'actions qui seront détectés par la couche numérique. Le même cycle se répète jusqu'à la fin de la simulation.

En comparaison avec PAX1, le chemin tracé par la boucle de simulation est réduit d'une couche. De plus, le caractère purement matériel des couches impliquées dans la simulation minimisent les délais de communication et favorisent un calcul parallèle intensif. Cependant, le système perd en termes de flexibilité et de souplesse d'implémentation en comparaison avec une réalisation logicielle. Ces pertes sont minimisées par l'utilisation des circuits électroniques programmables (FPGA) accompagnés d'outils de synthèse et de routage automatiques des descriptions matérielles.

L'avantage d'une implémentation purement matérielle est la réduction des latences de communication avec la machine hôte. Le calcul matériel de la plasticité libère le système d'une connexion permanente avec la machine hôte. Il devient donc possible d'élargir la taille du réseau en profitant de la rapidité du traitement et du parallélisme du calcul matériel. Bien que dans cette version le nombre de neurones soit limité à 25, la taille maximale du réseau pourrait être de quelques centaines (voir Chapitre 3, section 4 pour une étude détaillée des performances du système). Il existe, cependant, des limites liées à la programmation embarquée ; les ressources matérielles, telles que les mémoires et les unités de calcul, deviennent moins disponibles en taille et en nombre (Belhadj, 2009b). Le réalisateur est amené à compenser ce manque de ressources en optimisant les implémentations. Ceci peut conduire à des réalisations peu lisibles mais néanmoins astucieuses.

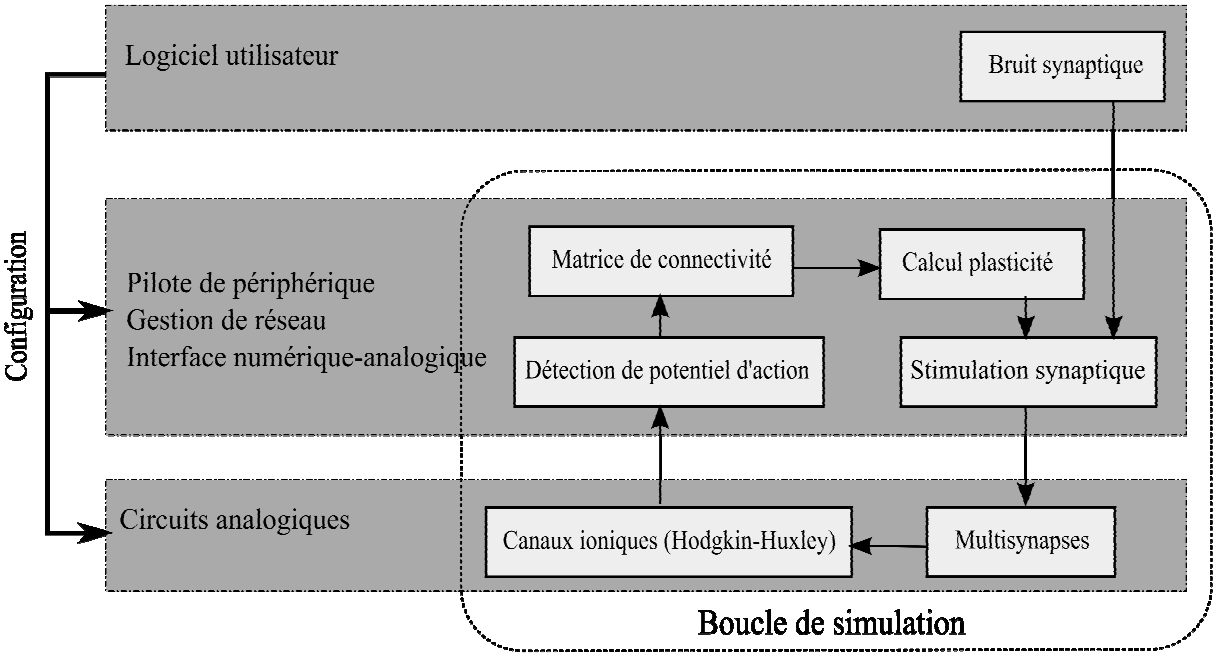



Figure 2.7. Répartition des calculs dans les couches de PAX2.

### 4.2 La carte *Gaillimh*

La carte *Gaillimh*[1] accueille 5 ASICs nommés *Galway*[2] et un FPGA *Xilinx Spartan 3 XC3S1500FG456*. Chaque ASIC contient 5 neurones. Le nombre total de neurones est donc de 25. Chaque neurone est directement connecté au FPGA. Un port série RS232 permet la liaison du système avec la machine hôte. Tout le système est cadencé avec une horloge de 100 MHz. La figure 2.8 fournit une photographie annotée de la carte.

Vu le nombre modeste des ASICs, le FPGA est directement connecté à chacun d'entre eux. Des convertisseurs numérique/analogique (DAC) transforment les paramètres de configuration envoyés par le FPGA en des valeurs analogiques interprétables par l'ASIC. Deux mémoires SDRAM de taille 256 Mo chacune sont accessibles à partir du FPGA. Elles peuvent servir pour le stockage momentané des résultats de simulation.

Comme le montre la figure 2.8, chaque ASIC possède des broches de tests pour visualiser les signaux clés de l'ASIC et pour injecter du courant de stimulation. Ces broches sont utilisées pour le calibrage des paramètres analogiques à l'intérieur du circuit. Ils servent à stimuler les neurones lorsqu'ils sont mis en mode *voltage clamp* ou à visualiser l'état des canaux ioniques.

Les paramètres destinés à la configuration des neurones analogiques ont une adresse composée de trois informations : le type de paramètre (analogique ou numérique), le numéro de l'ASIC cible et le numéro du neurone dans l'ASIC. Les paramètres numériques sélectionnent les conductances qui seront mises en jeu par le modèle de neurone, alors que les paramètres analogiques règlent les propriétés intrinsèques des conductances.

A côté du paramétrage des neurones, il y a aussi le paramétrage du réseau qui consiste à la configuration de la matrice de connectivité et les composants de calcul de la plasticité. Il s'agit de définir les listes de neurones pré et postsynaptiques de chaque neurone et de spécifier la nature des connexions (excitatrices ou inhibitrices). Le protocole de configuration utilisée pour cette tâche est décrit en détail dans l'annexe A.

Le bruit synaptique est une application logicielle qui génère selon une distribution statistique des intervalles de temps d'injection d'un stimulus dans chaque synapse en simulant l'activité synaptique corticale environnante. Encore une fois, le FPGA reçoit ces informations, extrait les adresses et transmet l'information utile sur la(es) synapse(s) cible(s). L'injection du bruit synaptique s'ajoute à la valeur des poids synaptiques au niveau de la multisynapse. Les détails de cette application font l'objet de la section 3 du Chapitre 3.

[1] Gaillimh est le nom d'une ville en Irlande du nord. Le droit d'attribution des noms aux cartes est attribué au réalisateur (C. Lopez).
[2] Galway est le nom d'une ville en Irlande. Le droit d'attribution des noms des circuits est attribué au concepteur (Y. Bornat)

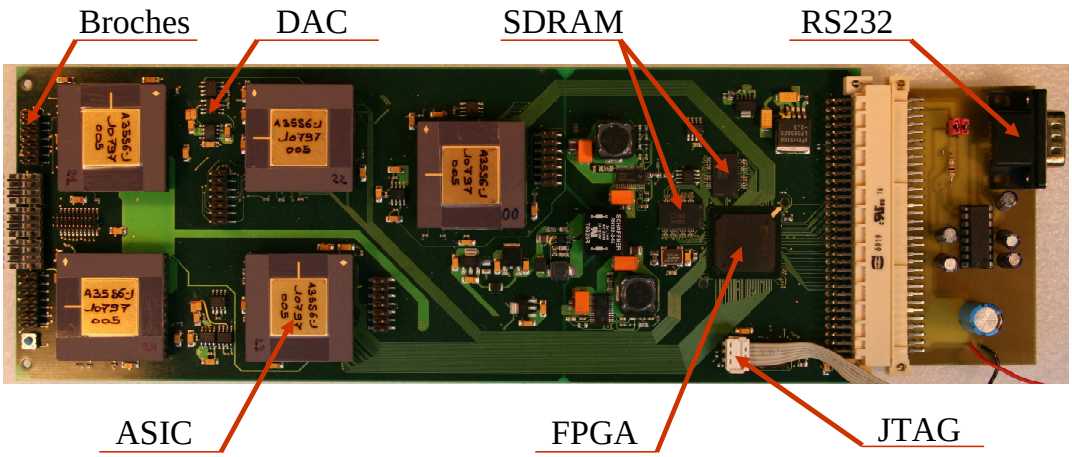



Figure 2.8. Photographie de la carte *Gaillimh*

### 4.3 Le circuit *Galway (Bornat, 2006)*

A la différence du circuit *Trieste*, le circuit *Galway* est conçu pour simuler des modèles de neurones configurables. Il s'appuie sur les travaux de thèse de S. Saïghi (Saïghi, 2004) qui ont abouti à la réalisation d'une bibliothèque de fonctions analogiques. La bibliothèque a été réalisée en technologie BiCMOS 0.35µm chez *austria*micro*systems*. Elle rassemble les opérations nécessaires à l'intégration d'un modèle de neurone qui suit le formalisme de Hodgkin-Huxley. Le circuit Galway occupe 10.5 $mm^2$ pour 50000 composants actifs et passifs.

La conception d'un système contenant plus de neurones nécessite d'intégrer plus de neurones par circuit. *Galway* contient 5 neurones entièrement configurables. Il existe deux types de paramètres de configuration ; les paramètres analogiques pour le réglage des courants ioniques, et les paramètres numériques pour la sélection des conductances appartenant à un même neurone. Un neurone peut avoir entre 3 et 5 conductances (ou canaux ioniques). Dans *Galway*, le neurone 0 ne peut avoir que 3 conductances, il servira comme neurone inhibiteur de type FS (*Fast Spiking*). Les autres neurones sont considérés comme excitateurs de type RS (*Regular Spiking*) et peuvent incorporer 4 ou 5 canaux ioniques, à l'exception du dernier qui peut avoir une conductance de plus. Le nombre de neurones excitateurs et inhibiteurs dans le circuit reflète les proportions dans le néocortex, à savoir respectivement 80% et 20%.

Le schéma bloc d'un neurone dans le circuit *Galway* est similaire à celui de *Trieste*, à l'exception du module qui gère les paramètres analogiques. Ce module est maintenant contrôlable par l'utilisateur. Chaque ASIC possède 204 paramètres analogiques qui correspondent notamment aux paramètres de modèles de conductances. Ces paramètres sont stockés dans des mémoires analogiques et subissent un rafraîchissement toutes les 2 ms pour garder leur valeur. La figure 2.9 illustre la structure du circuit à partir d'une photographie annotée. La proportion de surface occupée par les mémoires analogiques se situe dans le cadre en haut à droite.

Les entrées synaptiques fonctionnent selon le même principe que les multisynapses du circuit *Trieste*. Le poids synaptique est toujours codé par la durée de l'impulsion de stimulation. Cette durée ne doit cependant pas dépasser 60 µs et peut être codée avec une précision de 13 bits pour un circuit de contrôle fonctionnant à 100 MHz. En parallèle, il est aussi possible d'ajuster la montée du courant synaptique en agissant sur la configuration des paramètres analogiques de la synapse avec une faible, moyenne ou grande contribution.

Il est important de rappeler que la durée de l'impulsion de stimulation est la somme des valeurs de plusieurs poids synaptiques. Pour éviter de dépasser la durée de remontée maximale (60 µs), le codage d'un poids synaptiques ne doit pas dépasser une certaine valeur inversement proportionnelle au nombre de neurones dans le réseau. Autrement dit, si un réseau contient N neurones, la valeur maximale du poids synaptique est 60/N µs. Au-delà de ce temps, des effets de saturation peuvent apparaître.

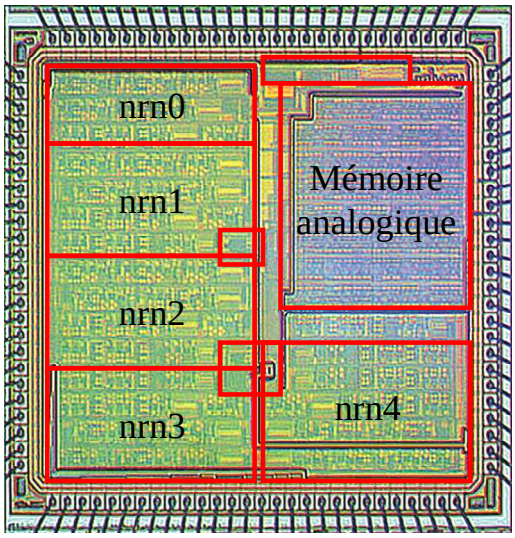



Figure 2.9. Photographie du circuit *Galway*

## 5 PAX3 : le système multicarte

Cette version de PAX a fait l'objet de la principale contribution de cette thèse. Le nombre de neurones possibles est de l'ordre de quelques centaines. La démarche utilisée consiste à renforcer le calcul parallèle en distribuant les neurones sur plusieurs cartes électroniques qui communiquent entre elles en temps réel. Le calcul neuronal reste purement matériel en vue d'accélérer le traitement et de réduire les latences.

### 5.1 Vue d'ensemble

Les principales caractéristiques de cette version sont le nombre élevé des cartes électroniques et la gestion distribuée du réseau. Les contraintes temps réel, précision de calcul et flexibilité de configuration sont également satisfaites dans cette version. Le nombre de neurones intégrés peut atteindre les 400.

On garde toujours la décomposition en trois couches dans cette version. La couche logicielle s'occupe de la génération et l'envoi des paramètres de configuration. Les deux couches matérielles sont subdivisées verticalement en plusieurs sous-unités. Chaque sous-unité représente une carte électronique constituée de circuits analogiques et un circuit numérique (FPGA), à l'exception d'une seule carte, dite carte mère, qui contient un circuit numérique et toute la circuiterie nécessaire pour l'interfaçage avec le logiciel utilisateur. Comme son nom l'indique, la carte mère pilote les autres cartes ″filles″ et assure leur interfaçage avec la machine hôte. L'ensemble des cartes filles constituent le réseau de neurones global. Chaque carte fille est responsable de la gestion de son propre sous-réseau interne et prend en charge les demandes d'interactions externes venant des autres cartes filles. Toutes les cartes sont connectées entre elles par l'intermédiaire d'un bus numérique (voir figure 2.10).

Du fait qu'il existe plusieurs cartes dans le système, il y a autant de boucles de simulation que de cartes filles. Le cycle de chaque boucle de simulation est similaire à celui du système PAX2, à la différence de la source de provenance des potentiels d'actions. Dans le système PAX2, les évènements proviennent uniquement des neurones locaux à la carte, alors qu'ici, ils peuvent provenir de neurones des autres cartes.

Les boucles de simulation s'exécutent en parallèle et ont la possibilité de communiquer entre elles. Un module de gestion de la communication externe est donc rajouté dans chaque carte. Ce nouveau mode de fonctionnement en réseaux distribués fait aussi appel à un calcul de plasticité distribué. La connectivité du réseau global est également gérée d'une façon distribuée. En général, chaque sous-unité doit être capable de gérer ses propres connexions locales, et les connexions externes afférentes. La section 3 du Chapitre 4 détaille l'approche utilisée pour la gestion des connexions.

Cette version bénéficie des avantages de l'implémentation purement matérielle des fonctions neuronales. En plus, la multitude des boucles de simulation renforce le calcul massivement parallèle et permet un plus grand nombre de neurones analogiques. Cependant, la communication inter-cartes induit des retards supplémentaires qui augmentent la latence dans les boucles de simulation (Belhadj, 2010). Le nombre de cartes que peut supporter le système est fonction de l'ampleur de ces retards. Ces derniers peuvent provenir de plusieurs sources : temps de propagation sur le bus, temps de traitement dans les dispositifs de routage, encapsulation/décapsulation des données, multiplexages, temps d'attente dans les buffers, etc. Pour garder les mêmes performances temporelles, il devient crucial de mettre en œuvre un protocole de communication temps-réel qui veille sur l'aspect temporel de l'échange de données inter-cartes. La section 2 du Chapitre 4 décrit la solution que l'on a développée pour garantir l'aspect temps réel du simulateur.

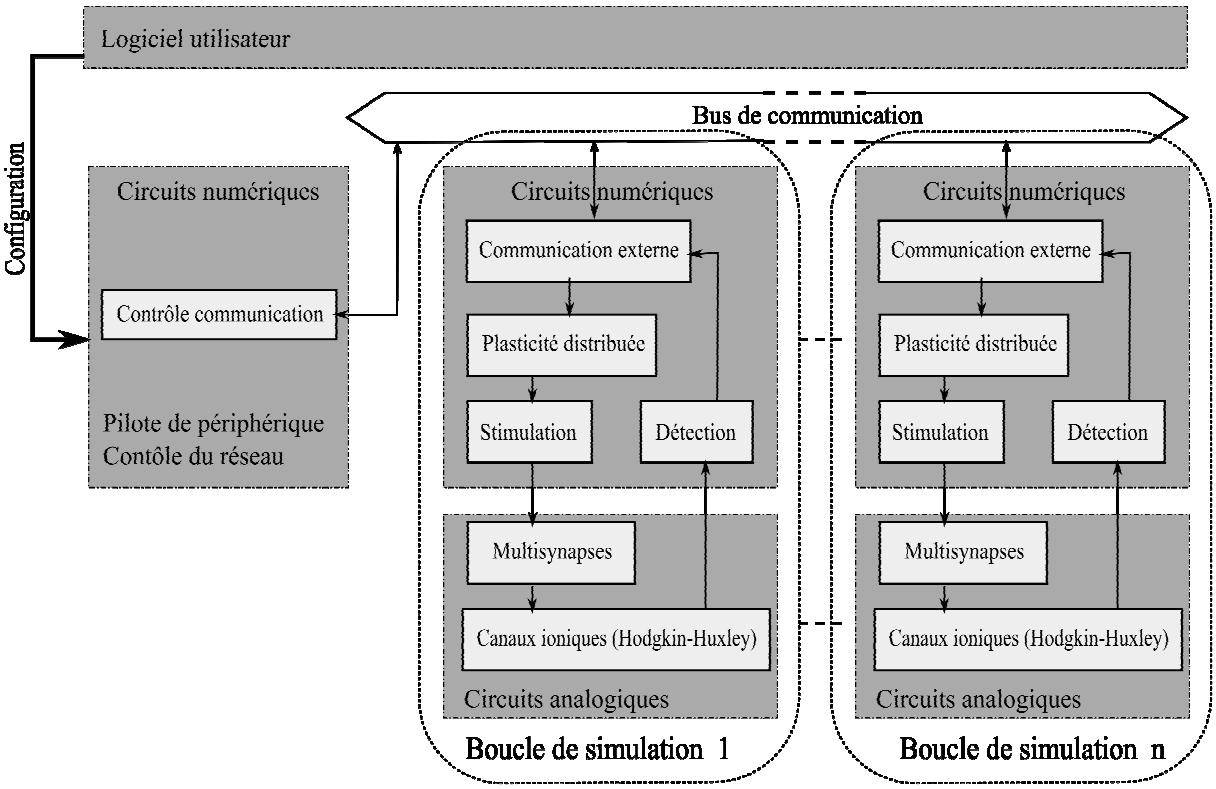



Figure 2.10. Répartition du calcul dans les couches de PAX3

### 5.2 La carte *Ekero*

La carte *Ekero*[1], ou carte fille, est une version améliorée de la carte *Gaillimh*. Cette dernière présentait le défaut de la nécessité d'une carte supplémentaire pour la génération des courants de stimulation, ce qui implique l'utilisation de générateurs externes de courants. La carte *Ekero* remplace l'utilisation de cette carte par l'utilisation des DACs[2] supplémentaires connectés directement aux ASICs pour commander les courants de stimulation à partir du FPGA. De plus, le routage sur la carte *Ekero* sépare le routage des composants analogiques du routage des composants numériques. Ceci permet de maîtriser les sources de bruits et de réduire la diaphonie[3]. Ces améliorations avaient comme prix l'élimination d'un ASIC à cause de la limitation de la surface disponible sur la carte *Ekero*. La figure 2.11 fournit une photographie annotée de la carte.

Le changement de la structure de la carte n'induit pas un changement dans le fonctionnement général de la boucle de simulation. Les mêmes circuits analogiques sont utilisés. Les mêmes principes d'interfaçage numérique-analogique sont repris dans cette version. Cependant, les protocoles de configuration et de communication changent à cause de la structure multicarte du système. L'entête des trames de configuration et de simulation est enrichie par le numéro de la carte cible pour spécifier le chemin de transit. L'annexe B décrit le protocole d'adressage des trames au sein du système PAX3.

La communication avec l'extérieur se fait par l'intermédiaire d'un connecteur spécial dont les entrées/sorties sont pilotés par le FPGA. Ce connecteur est directement lié au bus de communication situé sur le fond de panier. La programmation du FPGA se fait par l'intermédiaire de 3 entrées spécifiques sur le connecteur qui remplace le port JTAG traditionnel.

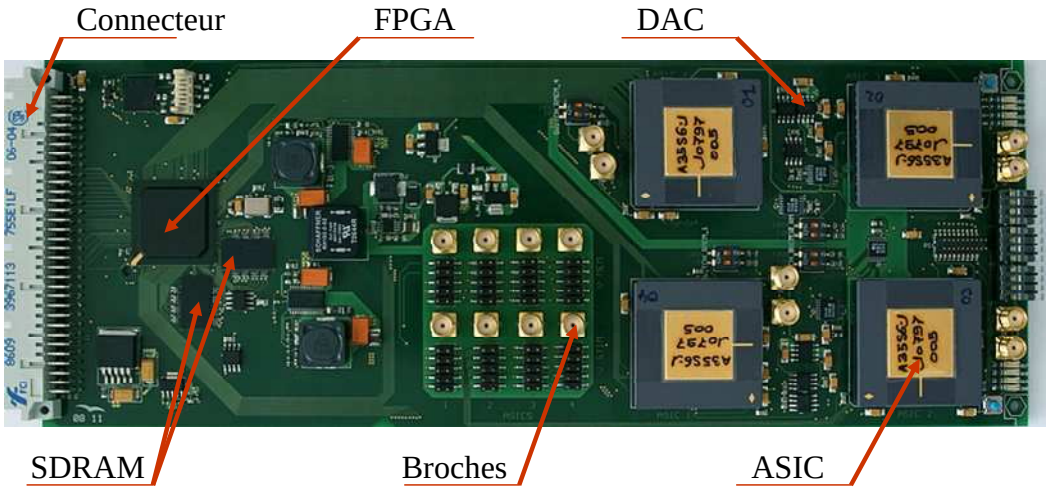



Figure 2.11. Photographie de la carte *Ekero*

[1] Ekero est le nom d'une ville au Suède. Le droit d'attribution des noms aux cartes est attribué au réalisateur (O. Malot).
[2] Digital to Analog Converter
[3] Interférence d'un premier signal avec un second, souvent à cause de phénomène d'induction électromagnétique.

### 5.3 La carte *Thalamos*

La carte *Thalamos*[1], ou carte mère, est le médiateur entre la machine hôte et le système. Elle accueille un FPGA de marque *Xilinx Virtex IV XC4VFX20*, un composant *FX2LP* pour la gestion du protocole de communication USB[2] et plusieurs types de connecteurs. Un connecteur spécial servira d'interface avec le bus de communication. Une extension du système est possible grâce aux broches de liaison disposées à l'extérieur de la carte. Deux mémoires de 256 Mo chacune sont prêtes à être pilotées par le FPGA en cas de besoin. Tout le système est cadencé par une horloge de 100 MHz. La photographie annotée de la carte est présentée dans la figure 2.12.

La carte mère possède plusieurs rôles dont les principaux sont :

- La gestion de la communication USB en mode bidirectionnel avec la machine hôte,
- La configuration des cartes filles (le protocole de configuration est décrit dans l'annexe B),
- Le contrôle de la communication sur le bus : lancement et arrêt de la simulation, détection des erreurs de simulation, identification des trames de contrôle, etc.,
- la programmation en chaîne des FPGAs des cartes filles.

Il est important de noter que la carte mère ne joue pas le rôle d'un *Maître* dans un système régi par une politique *Maître-Esclave*. L'évolution dans le système est spontanée. L'intervention de la carte mère par rapport au système ne dépasse pas un simple contrôle de la validité de trafic sur le bus en plus de la configuration des cartes filles.

D'un autre côté, la carte mère peut servir comme relais pour d'autres systèmes (neuromorphiques ou non). On peut imaginer une extension du système par une connexion de deux systèmes PAX3 identiques, où les deux cartes *Thalamos* constituent le relais. La taille des réseaux de neurones simulables double dans ce cas. Il est également possible d'interfacer des systèmes neuromorphiques développés par d'autres groupes de recherche ou un système d'interfaçage avec des cellules vivantes.

La carte mère sera enrichie dans sa prochaine version par des liaisons physiques haut débit. Ces améliorations entrent dans le cadre des efforts de standardisation des moyens de communication dans les systèmes neuromorphiques permettant de réaliser des simulations à partir des systèmes neuromorphiques hétérogènes.

[1] Thalamos est une déviation du terme Thalamus : une structure paire d'origine diencéphalique. Le droit d'attribution des noms aux cartes est attribué au réalisateur (C. Lopez).
[2] Universal Serial Bus : une norme relative à un bus informatique en transmission série.

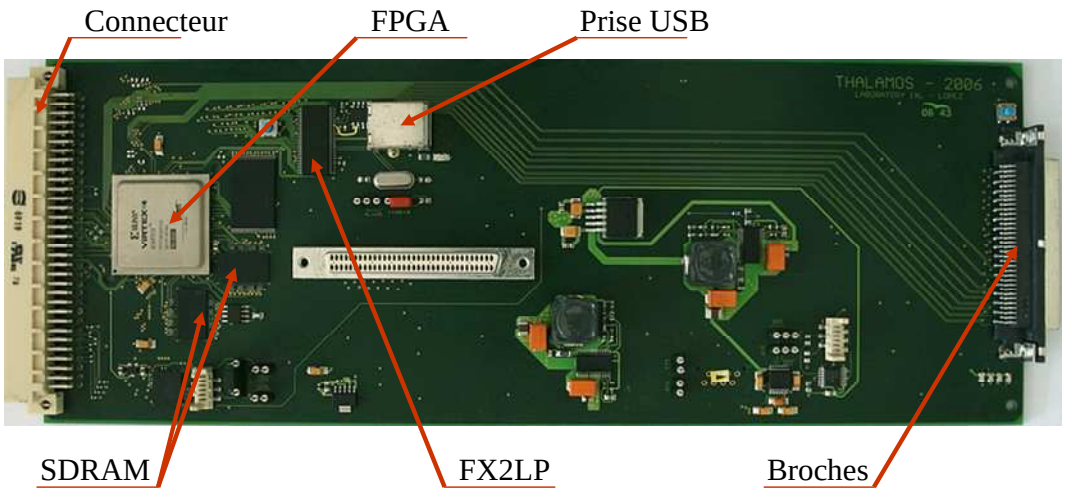



Figure 2.12. Photographie de la carte *Thalamos*.

### 5.4 Le système rack

Le système rack englobe toutes les cartes du système PAX3. Il dispose d'un fond de panier de technologie VME[1] pour connecter au plus 21 cartes au bus de communication. Le bus dispose de 68 fils parallèles disponibles pour l'envoi et la réception des données de communication, et de 5 fils spéciaux consacrés à la communication point-à-point entre les cartes (*daisy chain*). Un bloc d'alimentation placé dans le dos du rack fournit l'énergie électrique à toutes les cartes. La figure 2.13 est une photographie du système contenant 7 cartes : 6 cartes *Ekero* et une carte *Thalamos*.

Toutes les cartes ont un accès en lecture et en écriture sur le bus de communication. Le contrôle de l'accès est régi par un protocole de communication à base d'anneau à jeton : un signal sur 1 bit qui circule d'une carte à la suivante pour autoriser un envoi exclusif sur le bus. Le schéma bloc du système est donné dans la figure 2.14.

La transmission et la réception des données sur le bus se font d'une manière asynchrone pour garder l'aspect événementiel de la simulation. Les évènements sont les numéros des neurones qui ont généré des potentiels d'actions, associés au temps de génération. Toutes les cartes reçoivent une copie de tous les évènements qui transitent sur le bus. Un système de filtrage est placé à l'entrée de chaque carte fille pour rejeter les évènements qui n'ont pas de relation directe avec les neurones locaux.

[1] Une technologie de fabrication des bus de communication numériques.

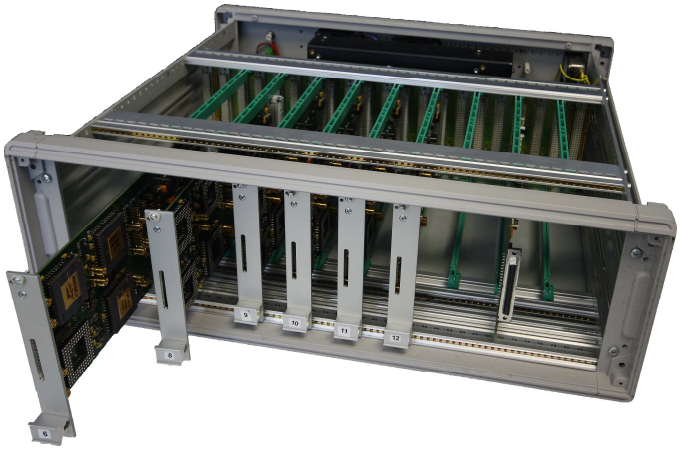


Figure 2.13. Photographie du système rack avec six cartes *Ekero* et une carte *Thalamos*.

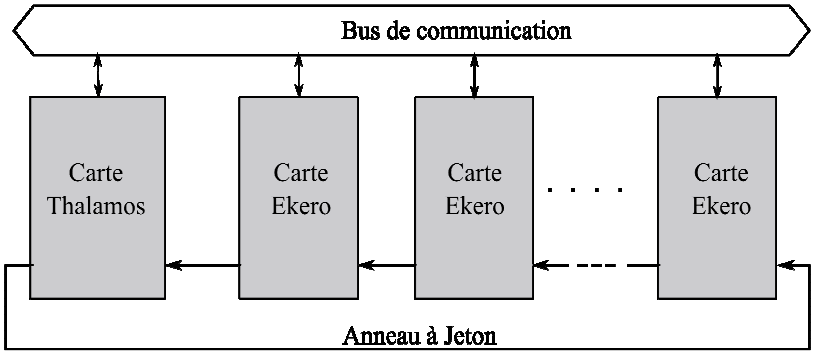



Figure 2.14. Schéma bloc du système rack.

Les évènements transmis sur le bus sont accompagnés par l'heure de leur apparition. Cette information est indispensable pour le traitement de la plasticité et l'interprétation des résultats de simulation. Le problème dans ce système vient du fait que les horloges locales des cartes ne sont pas synchronisées sur une même cadence. Ainsi, l'attribution des heures d'apparition aux évènements ne suit pas un temps absolu commun à tous les neurones. Il devient donc nécessaire de synchroniser les compteurs dans chaque carte. Cette opération est prise en charge par la carte mère qui incrémente périodiquement les compteurs internes de toutes les cartes filles.

## 6 Conclusion

Pour simuler des réseaux de neurones biologiquement réalistes, on a appliqué un certains nombre d'abstractions par rapport aux systèmes biologiques. Le principe d'un simulateur mixte analogique-numérique a été choisi. Les systèmes réalisés utilisent des circuits intégrés analogiques pour réaliser les neurones. Ces circuits simulent un nombre réduit de modèles figés basés sur le formalisme de Hodgkin-Huxley. La commande des circuits, et des convertisseurs de signaux numériques vers analogiques, est entièrement réalisée par un FPGA. Le système global est nommé PAX pour *Plasticity Algorithm Computing System*.

Plusieurs versions du système PAX ont été réalisées. La première version, ou PAX1, reposait sur une gestion logicielle du réseau. Un ordinateur fonctionnant avec un système d'exploitation temps-réel accède directement à chaque circuit d'une carte électronique. Il simule le comportement du réseau en lisant l'état des neurones, puis en envoyant les

stimulations correspondantes sur les entrées synaptiques appropriées. Le calcul de la plasticité des connexions synaptiques est une application logicielle. Malgré le fait que la composante logicielle apporte une grande flexibilité à la gestion du système global, elle ne permet pas d'envisager de réseau de taille supérieure à 20 neurones fonctionnant en temps-réel biologique.

La deuxième version de PAX, ou PAX2, remplace la composante logicielle par une gestion purement matérielle du réseau. Un FPGA contrôle la boucle de simulation dans tout le système. Il scrute les sorties des circuits, calcule la plasticité et stimule les neurones. Le caractère purement matériel de la boucle de simulation réduit la latence de la communication et du contrôle. Il devient possible de simuler des réseaux de taille supérieure à 200 neurones. Cependant, le système n'utilise qu'une seule carte dont la surface disponible ne permet pas d'intégrer plus que 25 neurones.

La troisième version de PAX, ou PAX3, multiplie le nombre de cartes dans le système, et les rassemble dans un même fond de panier. Le système global peut simuler jusqu'à 500 neurones avec des fonctions de plasticité, toujours dans les conditions du temps réel biologique. Cette architecture multicarte nécessite une gestion distribuée du calcul et un routage temps-réel de l'information entre des cartes.

Les chapitres 3 et 4 détaillent les deux dernières versions du système PAX aux quelles nous avons contribué. On mettra l'accent sur les techniques utilisées pour migrer vers une implémentation purement matérielle et les méthodes de quantification et de garantie du fonctionnement temps-réel.

# 3. PAX2 : le premier simulateur matériel

Le système PAX2 constitue la première contribution de cette thèse. Le but derrière la conception de ce système, dont le cahier des charges est présenté dans la section 1 de ce chapitre, est d'élargir la taille des réseaux de neurones utilisés dans les simulations. L'enjeu principal est de remplacer la gestion logicielle de l'ancien système (PAX1) par une gestion purement matérielle, ce qui permettra de réduire les latences et d'accélérer les calculs. Ce chapitre décrit les démarches et les techniques mises en œuvre pour effectuer cette migration.

Constitué d'une seule carte électronique, le système dispose d'un FPGA pour la gestion du réseau et de plusieurs ASICs pour simuler les neurones. Le FPGA accède directement aux circuits des neurones et effectue tous les calculs et les contrôles nécessaires pour créer une boucle de simulation fonctionnant en temps réel. Le noyau de la gestion du réseau est le calcul de la plasticité. La phase de configuration des différents composants du système est présentée dans la section 2. La section 3 développe les techniques utilisées pour la réalisation effective de la plasticité sur FPGA. Plusieurs approches de réalisation ont été explorées et comparées pour déduire la meilleure implémentation en termes de ressources matérielles et de puissance de calcul. La mise en réseau des circuits des neurones et leur interfaçage avec le FPGA font l'objet de la section 4.

## 1 Le cahier des charges

### 1.1 Une configuration flexible

Dans le système PAX1, la gestion logicielle du réseau simulé offrait une grande flexibilité dans la configuration de la topologie du réseau, des types de neurones et des paramètres du calcul de la plasticité. Dans le système PAX2, on souhaite migrer vers une implémentation purement matérielle tout en gardant cette flexibilité de configuration et de paramétrage des composants.

La programmation logicielle est remplacée par la configuration d'un FPGA. Pour faciliter la tâche, on a utilisé un environnement de développement spécifique au composant permettant de décrire l'implémentation numérique en langage VHDL[1]. Des outils de synthèse, de placement et de routage transforment la description matérielle en une configuration finale spécifique au FPGA cible.

[1] Le langage VHDL (Very high speed integrated circuit Hardware Description Language) est un langage de description matérielle destiné à représenter le comportement ainsi que l'architecture d'un système électronique numérique.

La réalisation d'un protocole de configuration est donc nécessaire pour router les valeurs des paramètres vers les composants cibles. Le FPGA, le centre de toutes les opérations de configuration, reçoit les trames de données RS232 de la machine hôte et les dirige soit vers les circuits analogiques soit vers d'autres modules de calcul définis en son sein. L'annexe A détaille le fonctionnement et l'utilisation du protocole de configuration.

### 1.2 Une plasticité en matériel

La carte du système PAX2 contient 5 ASICs, chacun héberge 5 neurones analogiques. Avec un total de 25 neurones, on sera amené à gérer jusqu'à 625 ($25^2$) connexions plastiques en temps réel. Cependant, notre but ici ne s'arrête pas au calcul de la plasticité des connexions, mais inclut aussi l'exploration de plusieurs réalisations qui permettent une implémentation optimisée en termes de ressources matérielles, précision de calcul et simplicité de mise en œuvre. La gestion de 625 connexions plastiques n'est que le minimum requis pour cette version, un nombre plus grand peut être atteint.

D'après l'algorithme de plasticité décrit par l'équation (1.12) (que l'on adoptera dans notre réalisation), une connexion plastique requiert le calcul de 4 exponentielles décroissantes en plus de 4 opérations de multiplications et 2 additions/soustractions. Etant donné que l'environnement matériel est limité en termes de ressources, des optimisations doivent être apportées à la programmation embarquée sur FPGA. Une approche de multiplexage des ressources de calcul est importante pour une réalisation économique.

### 1.3 Les contraintes temps-réel et précision de calcul

Les circuits analogiques simulent le comportement des neurones en temps continu et à l'échelle temporelle biologique (temps réel biologique). Pour suivre ce rythme de calcul, la plasticité du réseau doit également s'opérer en temps réel biologique.

Un poids synaptique est utilisé pour activer les synapses des neurones postsynaptiques. Sa valeur est mise à jour selon les règles de plasticité suite à chaque génération d'un évènement de part et d'autre des neurones pré et postsynaptiques. Dans le cas où la mise à jour du poids ne se fait pas à temps, la prochaine stimulation risque d'utiliser l'ancienne valeur du poids au lieu de la nouvelle. La simulation risque d'être altérée.

Pour remédier à ce risque, on définit une contrainte temps-réel pour le calcul de la plasticité : entre l'instant où un neurone génère un potentiel d'action et l'instant où il s'apprête à générer un deuxième, toutes les mises à jour des poids synaptiques relatives à ce neurone doivent être effectuées.

Une deuxième contrainte s'ajoute à ce niveau : la contrainte de la précision du calcul. Dans le système PAX2, on prêtera attention à la précision du calcul des poids synaptiques et à la finesse de la résolution temporelle afin de garantir le respect de la dynamique des modèles.

La variation des poids synaptiques au cours du temps permet d'interpréter les phénomènes de plasticité et d'apprentissage durant une simulation. Une représentation 16 bits des poids synaptiques est suffisante pour illustrer cette variation.

La résolution temporelle, quand à elle, est de l'ordre de la microseconde. Cette résolution a été choisie pour illustrer l'activité des mécanismes qui s'opèrent à l'échelle de la milliseconde. Il devient par conséquent possible de couvrir tous les changements majeurs qui peuvent se produire durant une simulation.

### 1.4 Un fonctionnement dans le pire cas

On appelle un pire cas dans un réseau de neurones impulsionnels, le scénario qui mène à la génération du plus grand nombre de potentiels d'action durant un intervalle de temps court.

Ceci a pour conséquence d'inonder les composants de calcul et de stockage par un grand nombre d'évènements. Le but derrière la définition de ce scénario, est de tester le fonctionnement du système dans les conditions extrêmes. Ainsi, on peut conclure sur sa validité de fonctionnement et/ou sur ces performances.

En général, le pire cas correspond au moment où tous les neurones génèrent simultanément des potentiels d'action dans un réseau complètement connecté (réseau *all-to-all*) et plastique. Les neurones sont également supposés utiliser leur taux de génération des potentiels d'action le plus élevé.

Dans le cas du système PAX2, le nombre de neurones est de 25. Dans les conditions du pire cas, on aura 25 potentiels d'action qui arrivent simultanément[1] au FPGA. Supposons que les neurones émettent des évènements avec un taux maximal de 100 évènements/s. Sachant que chaque évènement engendre le calcul des connexions pré et postsynaptiques, chaque potentiel d'action sollicite 50 calculs plastiques (25 connexions entrantes et 25 connexions sortantes). En considérant la contrainte temps-réel appliquée au calcul de plasticité, le FPGA doit faire face à 50*25 = 1250 calculs plastiques chaque période de 10 ms, soit 125 K opérations de calcul par seconde.

## 2 La configuration du système

Avant chaque simulation, le système reçoit une configuration. Cette dernière spécifie la connectivité du réseau (quel neurone est connecté à quel autre neurone) ainsi que la plasticité de ses connexions. Elle détermine également le type des neurones qui seront simulés et la polarité de leurs synapses (excitatrices ou inhibitrices).

Pour le système PAX2, on aura besoin de configurer le réseau (connectivité et plasticité), les ASICs et une fonction neuronale supplémentaire : le bruit synaptique.

### 2.1 Des matrices pour la configuration du réseau

Les connexions synaptiques sont à l'origine de la plasticité du réseau. Dans un réseau de N neurones, il existe $N^2$ connexions synaptiques possibles. Une connexion synaptique est caractérisée par les deux neurones qui la constituent (neurone pré et postsynaptique), sa plasticité (plastique ou non) et sa polarité (excitatrice ou inhibitrice).

Pour visualiser toutes les connexions synaptiques d'un réseau, la représentation matricielle est la plus adaptée. Une matrice carrée d'ordre N permet de représenter toutes les connexions synaptiques possibles dans un réseau constitué de N neurones. La figure 3.1 illustre un exemple de configuration d'un réseau de 3 neurones. Trois matrices d'ordre 3 sont définies. Les lignes représentent les neurones postsynaptiques et les colonnes représentent les neurones présynaptiques. Un élément de la matrice est référencé par une ligne et une colonne, ou encore par les coordonnées (*pre*,*post*) (Belhadj, 2008b).

[1] Deux potentiels d'action sont considérés simultanés s'ils sont reçus par le FPGA dans l'intervalle d'une microseconde (unité de la résolution temporelle).

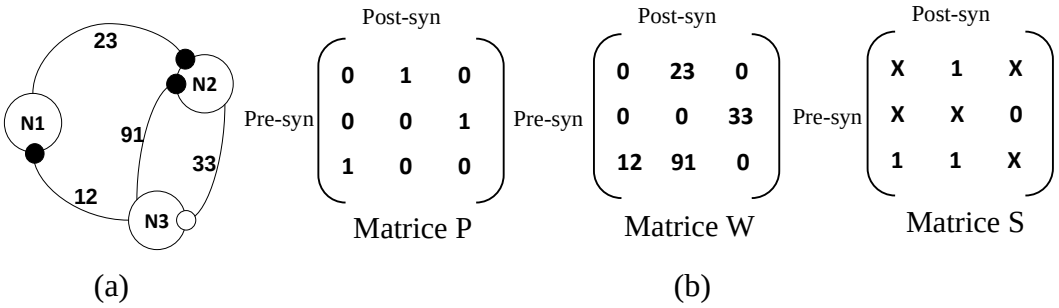



Figure 3.1. Représentation matricielle des connexions synaptiques. (a) La spécification d'un réseau composé de 3 neurones. (b) La représentation matricielle équivalente à la spécification du réseau. La matrice P représente la plasticité des connexions. La matrice W détermine les valeurs initiales des poids. La matrice S détermine la polarité des connexions.

La matrice P, ou matrice de plasticité, indique si une connexion est plastique (P(*pre*,*post*) = 1) ou non (P(*pre*,*post*) = 0). La matrice W, ou matrice des poids, contient les valeurs initiales des poids synaptiques. A partir de ces deux matrices, on peut déterminer si une connexion existe ou non. Dans le cas où P(*pre*,*post*) et W(*pre*,*post*) sont nuls, la connexion n'existe pas entre les neurones *pre* et *post*, sinon, elle existe. Une troisième matrice, appelée matrice S, indique si la connexion est excitatrice (S(*pre*,*post*) = 1) ou inhibitrice (S(*pre*,*post*) = 0). L'élément S(*pre*,*post*) est indifférent si la connexion n'existe pas.

Ces matrices traduisent donc la topologie du réseau, la plasticité des connexions et le type des interactions entre les neurones. Toutes les possibilités des connexions inter-neurones sont envisageables, en allant d'un réseau faiblement connecté à un réseau complètement connecté (réseau *all-to-all*). L'annexe A montre comment décrire ces matrices pour les insérer dans les blocs mémoires du FPGA.

### 2.2 Le bruit synaptique

Le bruit synaptique est une fonction neuronale supplémentaire utile pour créer un environnement de simulation proche de celui de la biologie. Elle permet de simuler l'influence de l'activité des neurones du voisinage. Cette influence, ou bruit, contribue à l'excitation ou l'inhibition des synapses des neurones impliqués dans une simulation et peut provoquer la génération précoce des potentiels d'action, ou au contraire, les annuler.

Pour les neurones corticaux, il a été montré que le bruit synaptique peut être approximé par une loi de Poisson (Abeles, 1982). Ce comportement peut être modélisé par une séquence de stimulations appliquée aux entrées synaptiques d'un neurone. Le bruit synaptique peut donc être assimilé à une variabilité des intervalles de potentiels d'action, ou encore *Inter-Spikes Interval* (ISI) en anglais (Calvin, 1967 ; Calvin, 1968).

Les travaux de thèse du Dr. A. Daouzli ont porté sur ce sujet et ont abouti au développement d'une application qui génère des séquences de bruit pour le système PAX2 (Daouzli, 2009). L'application tourne sur un ordinateur et génère des séquences d'ISIs pour chaque entrée synaptique des neurones du système. Ces séquences seront envoyées par la suite au FPGA où elles seront interprétées par un module spécifique. Les détails des trames de configurations du module du bruit synaptique sont également fournis dans l'annexe A.

### 2.3 La configuration des ASICs

La configuration des ASICs consiste à générer et envoyer les paramètres qui définissent le type de neurone. Le circuit utilisé pour simuler les neurones dans le système PAX2 est *Galway* (voir Chapitre 2, section 4.3). La configuration de *Galway* se fait par l'intermédiaire

de deux types de paramètres : les paramètres analogiques et les paramètres numériques (Bornat, 2007). Les paramètres analogiques servent à spécifier le comportement des conductances ioniques et à caractériser leur réponse à un courant de stimulation donné. Ces paramètres sont stockés dans des mémoires analogiques périodiquement rafraichies pour conserver leur valeur nominale. Les paramètres numériques sélectionnent les conductances à utiliser pour définir le type de neurone. A titre d'exemple, un neurone de type FS (*Fast Spiking*) nécessite 3 conductances pour modéliser les canaux potassique, sodique et de fuite, nécessaires pour reproduire sa dynamique.

Ces paramètres sont encapsulés et envoyés par la machine hôte au FPGA. Chaque paramètre est accompagné par l'adresse de l'ASIC destination et du neurone cible. Une opération de décapsulation et de décodage des adresses se fait au niveau du FPGA pour les router vers les ASICs correspondants. Les valeurs des paramètres analogiques passent par des convertisseurs numérique-analogique (DAC) avant d'être acheminés vers les mémoires spécifiques à l'intérieur des circuits.

Une fois stockées dans le circuit cible, les valeurs des paramètres analogiques doivent être continuellement rafraîchies à une période de 2 ms. Elles seront donc gardées dans le FPGA sous leur forme numérique et renvoyées aux DACs chaque 2 ms. L'annexe A fournit les détails de cette opération ainsi que les formats des trames utilisées pour la construction des mots de configuration.

## 3 Une gestion matérielle du réseau

Une fois la phase de configuration est achevée, la phase de simulation est prête à être lancée. Pendant la simulation, le FPGA interprète les données de configuration du réseau pour déterminer les calculs qu'il doit effectuer. Le résultat de ce calcul est par la suite utilisé pour stimuler les neurones postsynaptiques. Ce cycle est répété identique à lui même jusqu'à la fin de la simulation. Le calcul de la plasticité demeure donc le centre des opérations de la gestion du réseau. Il nécessite un soin particulier lors de sa conception pour optimiser son fonctionnement et éviter qu'il soit un goulet d'étranglement de l'opération de tout le système.

Dans cette section, on propose d'explorer des réalisations possibles d'un calcul de la plasticité sur un FPGA. Les réalisations seront par la suite comparées en termes de ressources matérielles requises et puissance de calcul pour enfin sélectionner la meilleure implémentation.

### 3.1 Le calcul de la plasticité sur FPGA

Cette partie détaille les techniques et les démarches utilisées pour la réalisation effective de l'architecture du calcul de la plasticité sur FPGA. On suivra une approche de bas-vers-haut (*bottom-up*) pour décrire les différentes étapes de réalisation.

Le modèle de plasticité que l'on implémentera est celui décrit par l'équation (1.12) rappelée ci-dessous :

$$\frac{dw_{ij}}{dt} = \varepsilon_i \ \varepsilon_j \ \{ \ (w_{LTP} - w_{ij}) \sum_{t_i} P\left[(t - t_j^{last}(t)\right] \delta(t - t_i) - (w_{ij} - w_{LTD}) \sum_{t_j} Q\left[(t - t_i^{last}(t)\right] \delta(t - t_j) \ \}$$

Dans ce modèle, le calcul d'une variation d'un poids synaptique utilise quatre fonctions à exponentielles décroissantes plus 4 multiplications et 2 additions/soustractions (Destexhe, 2004). Dans un premier temps, le but est d'arriver à effectuer le calcul sur FPGA d'une seule variation du poids, ensuite, de couvrir toutes les connexions du réseau.

Le FPGA cible est un Xilinx Spartan 3 XC3S1500FG456. Il contient 32 multiplicateurs dédiés, 32 blocs RAM de 18 Kbits chacun, 208 Kbits de mémoire distribuée, environ 30 K de cellules logiques et 487 pins d'entrée/sortie (Xilinx, Spartan3). Dès les premiers tests, les ressources requises pour le calcul d'une variation de poids synaptique et les ressources disponibles sur le FPGA indiquent que ce dernier ne peut supporter le calcul brut que pour quelques connexions plastiques. L'emploi des méthodes d'optimisation sont donc inévitables.

*Calcul d'une exponentielle décroissante*

Plusieurs méthodes existent pour le calcul d'une exponentielle sur FPGA. La méthode la plus couramment employée est celle qui utilise des valeurs précalculées de l'exponentielle et les stocke dans une table de Look-Up (ou table de correspondance). Une valeur est donc lue à chaque demande de calcul. Cette méthode souffre de défauts importants qui la rendent peu satisfaisante. En effet, les accès à une table s'effectuent séquentiellement. La mise en œuvre de calcul hautement parallèle nécessite plusieurs tables, ce qui consomme trop de mémoire en FPGA. D'un autre côté, une mise à l'échelle en amplitude et en évolution temporelle n'est pas possible avec cette méthode. Les valeurs précalculées ne couvrent qu'une gamme fixe des valeurs du cycle d'évolution d'une exponentielle. Cet inconvénient ne peut être évité que par un calcul permanent qui évolue en fonction du temps et de l'amplitude.

D'autres méthodes comme l'approximation polynômiale et l'utilisation d'un processeur mathématique peuvent remédier aux inconvénients de la première en réalisant des optimisations poussées du calcul. Par contre, l'inconvénient ici reste que le calcul est séquentiel, et ne peut être aisément parallélisé.

La méthode que l'on a adoptée pour le calcul des exponentielles est inspirée des concepts de calcul analogique. Elle a été développée et évaluée par le Dr. Y. Bornat (Bornat, 2006) qui a transformé une interprétation du calcul de l'exponentielle du domaine du continu au domaine discret. La méthode consiste à résoudre en temps réel l'équation différentielle dont l'exponentielle est solution,

$$f(t) = e^{-t} \Rightarrow f'(t) = -f(t) \tag{3.1}$$

Nous obtenons alors :

$$\frac{df}{dt}(t) = -f(t) \tag{3.2}$$

$$\frac{f(t+dt) - f(t)}{dt} = -f(t) \tag{3.3}$$

$$f(t+dt) = f(t) - f(t).dt \tag{3.4}$$

Dans le domaine numérique, la variation infinitésimale est considérée comme un quanta qui tend vers 0. La formule (3.4) devient dans ce cas :

$$f(t+\delta t) = f(t) - f(t).\delta t \tag{3.5}$$

Cette formule est très intéressante pour du calcul réalisé sur FPGA. En effet, si la fonction *f(t)* est mémorisée dans un registre sous forme binaire et que l'on choisi $\delta t = 2^{-n}$, la multiplication $f(t).\delta t$ s'obtient par décalage de n bits de *f(t)*, ce qui signifie que son implémentation ne nécessite aucun composant particulier. Elle ne nécessite qu'un registre et un soustracteur sur 2n bits. La figure 3.2 fournit le circuit de calcul de la formule (3.5). L'entrée *init* du registre permet d'initialiser la valeur de l'exponentielle pour t = 0.

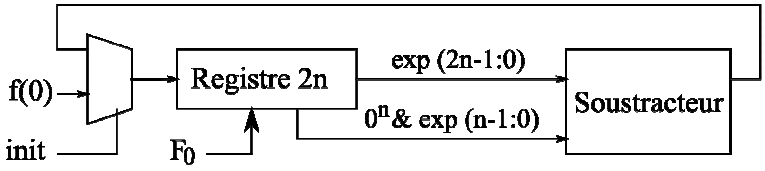



Figure 3.2. Circuit de calcul d'une exponentielle de la forme $e^{-t/\tau}$.

L'opération de calcul est cadencée par une fréquence de fonctionnement $F_0$ qui se détermine par la valeur de la constante de temps $\tau$. La dépendance entre la constante de temps et la fréquence de fonctionnement peut être déterminée par la méthode de l'intersection de la dérivée en 0 avec l'axe des abscisses. Considérons $P_0$ la période de la fréquence de fonctionnement, on a :

$$\tau . f'(0) = -f(0) \tag{3.6}$$

$$\tau = \frac{f(0)}{f'(0)} = P_0 . \frac{f(0)}{f(\delta t) - f(0)} \tag{3.7}$$

$$\tau = P_0 . \frac{f(0)}{f(0).(1-\delta t) - f(0)} \tag{3.8}$$

$$\tau = \frac{P_0}{\delta t} = \frac{P_0}{2^{-n}} = P_0 . 2^n \tag{3.9}$$

A titre d'exemple, pour n = 10 et $F_0$ = 1 Mhz, on obtient $\tau$ = 1,024 ms.

Cette méthode a été configurée dans un FPGA pour évaluation. Le calcul étant réalisé en utilisant des entiers binaires, il rentre dans le cadre des techniques dites à virgule fixe.

Tableau 3.1. Erreur de la méthode de calcul selon le nombre de bits utilisés.

| **n** | **Erreur relative à f(0)** | **en valeur LSB** |
|---|---|---|
| **6** | 1,6% | 1 LSB (0,977) |
| **7** | 0,8% | 1 LSB (0,989) |
| **8** | 0,4% | 1 LSB (0,974) |
| **9** | 0,2% | 1 LSB (0,997) |
| **10** | 0,1% | 1 LSB (0,998) |
| **12** | 0,03% | 1 LSB (0,999) |
| **14** | 60,9 ppm | 1 LSB (0,999) |

Le calcul n'est donc pas aussi précis qu'une valeur calculée en virgule flottante. Le tableau 3.1 nous montre l'écart maximal entre la valeur calculée selon cette technique et celle calculée par l'outil *Maple* (en virgule flottante).

Les erreurs de calcul observables sont particulièrement raisonnables compte tenu du faible besoin en ressources numériques. La constante de temps utilisée ici dépend de la fréquence de fonctionnement du circuit et de la précision souhaitée.

*Calcul de la somme des exponentielles décroissantes*

La formule (1.12) emploie une somme d'exponentielles pour illustrer l'interaction avec les évènements antécédents, une exponentielle pour chaque évènement. Il est évident que le calcul parallèle d'un nombre si élevé d'exponentielle est impossible, vu les ressources matérielles disponibles. Même si on utilise une technique de multiplexage pour réduire le nombre de circuits de calcul, il reste quasiment impossible de répondre aux exigences de la réalisation d'une somme.

Encore une fois, la solution vient de la propriété mathématique de la somme des exponentielles à constante de temps égales, illustrée par l'équation (1.8). De point de vue pratique, il suffit de calculer une seule exponentielle et d'additionner à chaque fois la valeur de l'amplitude initiale à la valeur courante de l'exponentielle.

La somme des exponentielles n'est employée que dans les fonctions *P* et *Q* du modèle de la plasticité. La réalisation revient donc à utiliser le circuit de calcul de la figure 3.3 avec une mise à jour de la valeur du registre 2n bits suite à l'arrivée d'un évènement. La nouvelle valeur du registre consiste en la somme de la valeur courante de l'exponentielle et l'amplitude de départ (*f(0)*).

Prenons l'exemple du calcul d'une potentiation, ou LTP. L'équation équivalente est donnée par :

$$LTP = \sum_{t_j} P\left[(t - t_j^{last}(t)\right].\delta(t - t_i) = \sum_{t_j} A_p \, exp\left[-(t - t_j^{last})/\tau_p\right].\delta(t - t_i) \qquad (3.10)$$

Cette équation équivaut à la réalisation d'une somme des exponentielles déclenchées par l'arrivée des évènements présynaptiques. D'après la propriété énoncée par l'équation (1.8), la somme des exponentielles de la potentiation revient à ajouter l'amplitude $A_p$ à la valeur de l'exponentielle enregistrée à l'instant d'apparition de l'évènement. Cette opération se répète jusqu'à la fin de la simulation, ce qui permet de garder les traces de tous les évènements du neurone sans avoir besoin de les stocker. La valeur de LTP n'est saisie qu'à l'arrivée d'un évènement postsynaptique. La figure 3.3 illustre de concept sur une LTP à $t_i$ provoquée par 4 évènements présynaptiques à $t_{j1}$, $t_{j2}$, $t_{j3}$ et $t_{j4}$.

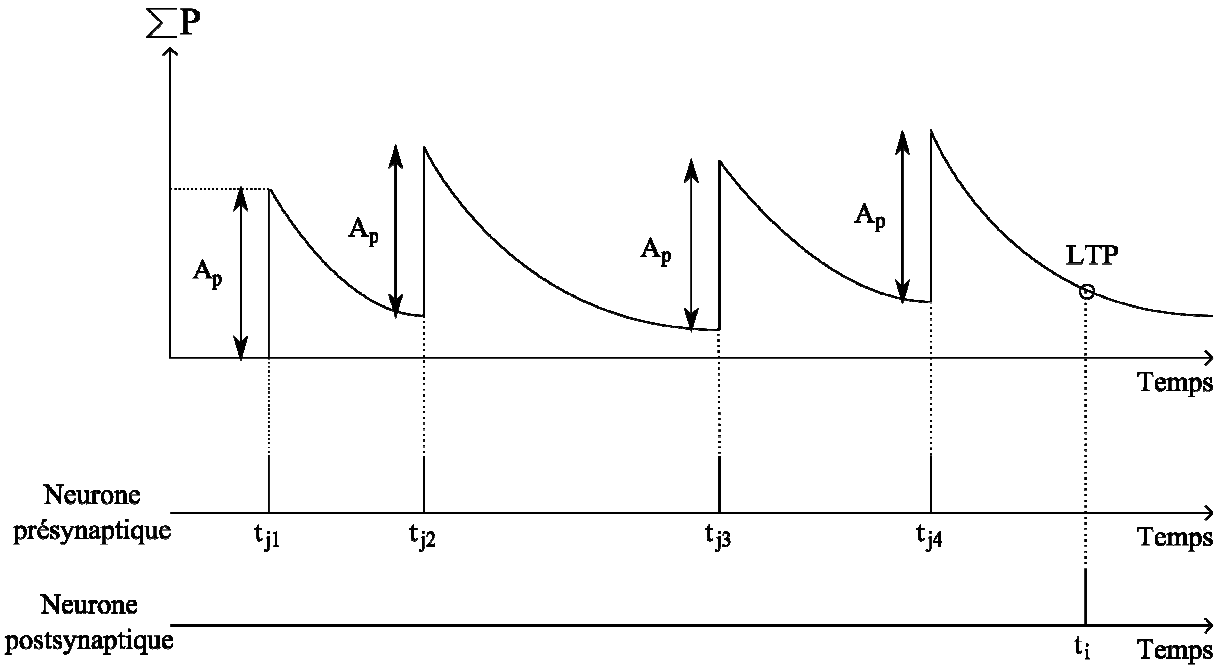



Figure 3.3. La somme des exponentielles de la fonction P.

*La réalisation du circuit de calcul d'une connexion plastique*

Après avoir déterminé le circuit qui implémente l'exponentielle et la somme des exponentielles, il ne reste qu'à déterminer le circuit STDP qui met en œuvre l'équation (1.12). Le schéma le plus intuitif est illustré dans la figure 3.4. $E_j$ et $E_i$ représentent respectivement les entrées des évènements pré et postsynaptiques. L'ordre d'arrivée des évènements sélectionne le calcul à effectuer, une LTP ou une LTD. Les valeurs des fonctions P et Q dépendent de l'intervalle de temps qui sépare les évènements pré et postsynaptiques. La variation de poids $\Delta w_{ij}$ est obtenue après multiplication par le facteur de saturation ($w_{LTP}$-$w_{ij}$ ou $w_{ij}$-$w_{LTD}$) et les facteurs de suppression ($\varepsilon_j$ et $\varepsilon_i$).

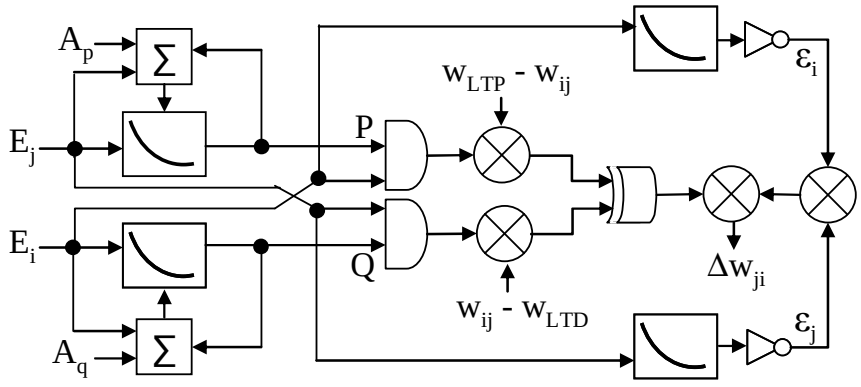



Figure 3.4. Circuit STDP de la réalisation d'une connexion plastique.

En plus des ressources matérielles utilisées par les blocs des exponentielles, le circuit STDP utilise 4 multiplieurs et deux soustracteurs. Le nombre de multiplieurs peut baisser à un seul en multiplexant temporellement son utilisation.

Le calcul d'une variation d'un poids dure 10 cycles d'horloge. La taille des registres 2n des blocs de calcul des exponentielles est de 16 bits (n = 8). Pour conserver cette précision dans les calculs subséquents, les registres utilisés pour stocker les opérandes des opérations de multiplication et de soustraction possèdent la même taille.

Ce circuit réalise une seule connexion plastique. Pour couvrir la totalité du calcul de la plasticité du réseau, plusieurs approches peuvent être utilisées. Dans la suite de cette section, on exposera trois réalisations possibles faisant appel à des approches différentes : l'approche parallèle, l'approche séquentielle et l'approche *orientée-cellule*. Dans la section d'après, on procédera à une comparaison des performances de ces approches et on choisira la plus adaptée à l'implémentation de la plasticité pour le système PAX2.

*Calcul de la plasticité du réseau : approche parallèle*

L'approche parallèle consiste à placer autant de circuits STDP que de connexions plastiques requises pour un calcul. La conception d'une telle architecture est simple et intuitive ; le circuit STDP est dupliqué identique à lui-même plusieurs fois. Chaque circuit STDP est connecté aux deux sorties de deux neurones constituant une connexion. Un neurone peut être connecté à lui-même. Les poids nouvellement calculés sont transmis aux blocs des multisynapses où ils seront transformés en courant synaptique. Le schéma bloc de la réalisation parallèle est fourni par la figure 3.5.

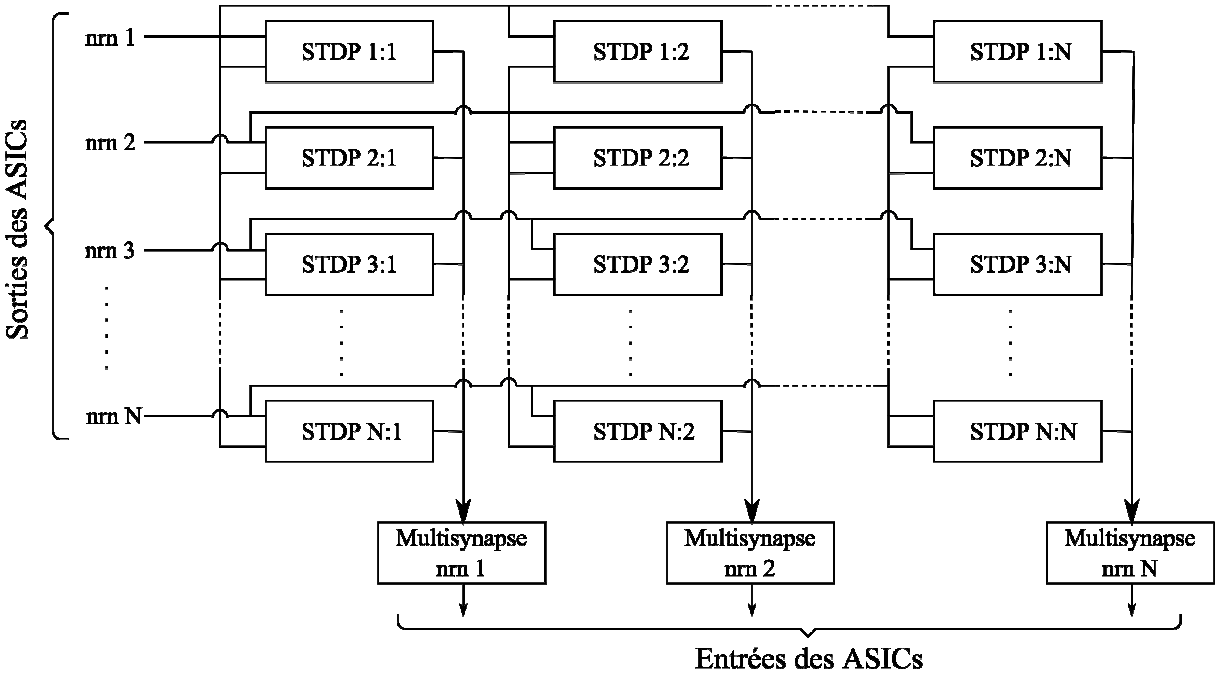



Figure 3.5. Illustration de l'approche parallèle pour le calcul de la plasticité du réseau : schéma bloc de la réalisation.

Les circuits STDP sont disposés sous forme de matrice d'ordre N, où N est le nombre de neurones dans le réseau. Les sorties des circuits d'une même colonne sont connectées à une même multisynapse. Cette réalisation est simple et rapide à concevoir et à mettre en œuvre.

Le calcul de la plasticité est massivement parallèle, ce qui induit des meilleures performances en termes de temps de traitement. La contrainte temps-réel est largement satisfaite dans ce cas : le calcul d'un poids synaptique prend 10 cycles d'horloge dans toutes les connexions. Sachant que le FPGA est cadencé par une horloge de 100 Mhz, la durée de

calcul de la variation d'un poids ou de tous les poids reste la même, soit 100 ns, ce qui reste très inférieure à 10 ms (valeur limite autorisée par la contrainte temps-réel).

Le câblage direct des circuits STDP aux entrées/sorties des ASICs remplace la configuration de la connectivité du réseau, puisque les connexions existent physiquement et sont activées ou désactivées par la présence ou non d'évènements aux sorties des ASICs. Cependant, la plasticité et les poids initiaux des connexions ne sont pas déterminés et nécessitent d'être configurés dans tous les circuits STDP de l'architecture.

Malgré le fait que la conception et la réalisation de l'approche parallèle sont simples et intuitives, elles souffrent cependant d'un inconvénient majeur qui limite ses chances d'être exploitable. En effet, cette réalisation nécessite beaucoup de ressources matérielles dont les plus critiques sont les multiplieurs. Pour réaliser les 625 connexions du système PAX2, il faudra au moins 625 multiplieurs, ce qui n'est généralement pas supporté par les FPGAs, même les plus puissants.

Nous avons tout de même réalisé sur FPGA cette approche. Le but était d'explorer les limites du FPGA que l'on utilise et de quantifier la puissance de calcul qu'il offre. Il sera possible par la suite de comparer cette implémentation à d'autres qui tendent à économiser les ressources matérielles. Cette réalisation servira à titre de comparaison et ne sera pas retenue comme une implémentation possible de la plasticité du réseau du système PAX2.

*Calcul de la plasticité du réseau : approche séquentielle*

Pour remédier à l'inconvénient de l'approche parallèle, on a procédé à la sérialisation du calcul. Les sorties des ASICs sont donc scrutées les unes après les autres et les potentiels d'action détectés (PAs) sont stockés dans une FIFO. Cette opération permet d'enchainer un calcul séquentiel. Désormais, cette approche sera nommée l'approche *séquentielle*.

Les potentiels d'action se transforment en évènements qui attendent leur tour de traitement dans la FIFO. Le calcul séquentiel consiste à tirer un seul élément de la FIFO à la fois, effectuer le calcul qui lui est relatif et transmettre le résultat au module de la multisynapse pour activer les entrées synaptiques des neurones cible. Ensuite, le même traitement est effectué pour l'évènement d'après, et ainsi de suite.

Cette façon de voir les choses implique une conception d'une architecture plus compliquée que l'architecture de la réalisation parallèle. Pour gérer le déroulement des opérations séquentielles, une machine d'état est nécessaire pour ordonnancer les traitements. Cette machine d'état, appelée aussi *ordonnanceur*, utilise les données de configuration en plus de l'évènement incident pour déterminer le calcul à effectuer. Ce dernier est réalisé par un circuit STDP spécifique. La figure 3.6 illustre l'architecture séquentielle proposée.

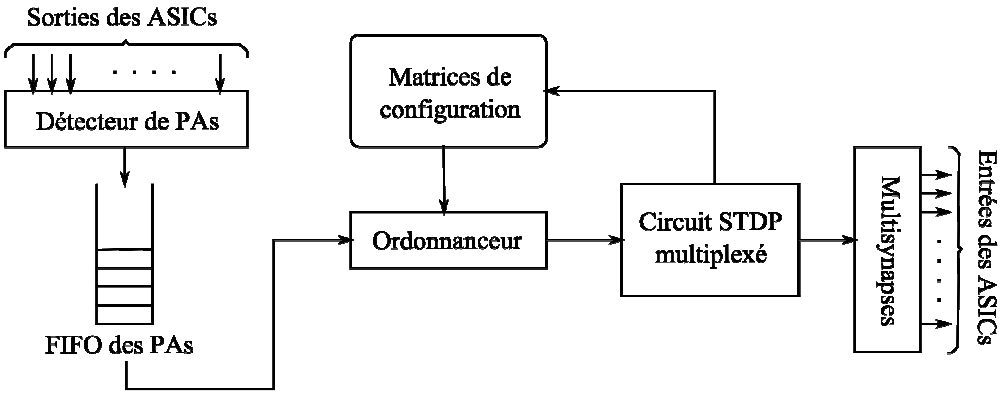



Figure 3.6. Schéma bloc d'une réalisation séquentielle de la plasticité du réseau.

Le calcul de la plasticité proprement dite est réalisé par un circuit STDP multiplexé dans le temps. Le multiplexage temporel permet de réduire les ressources matérielles nécessaires pour la réalisation des multiplieurs. Cependant, l'économie apportée aux ressources se fait au détriment du temps de traitement.

Le principe du multiplexage temporel consiste à diviser le domaine temporel en créneaux de longueur fixe. Chaque créneau est associé à une connexion. Le calcul d'une connexion plastique ne se déclenche que pendant le créneau de temps qui lui est alloué. Le premier créneau du temps est utilisé pour calculer la première connexion, le deuxième pour le calcul de deuxième connexion, etc. Lorsqu'on atteint le dernier créneau et la fin du calcul de la dernière connexion, on reboucle sur la première. Le même cycle se répète jusqu'à la fin de la simulation.

Le multiplexage du circuit STDP revient à répartir les ressources de calcul des fonctions P, Q, $\varepsilon_j$ et $\varepsilon_i$ sur plusieurs connexions, ce qui revient à multiplexer le calcul de leurs exponentielles. D'après l'équation (3.9), le rafraîchissement de la valeur de l'exponentielle se fait chaque période $P_0$. Cette dernière est désormais appelée période de rafraîchissement. Elle est proportionnelle à la constante de temps $\tau$. Etant donné que les valeurs des constantes de temps rencontrées dans le modèle de plasticité sont de l'ordre de la dizaine de millisecondes (voir tableau 1.1), la période de rafraîchissement est de l'ordre de la dizaine de microsecondes (en considérant le cas où n = 8). Sachant qu'un cycle d'horloge de le FPGA dure 10 ns, il devient possible d'utiliser le même circuit pour calculer plusieurs exponentielles sans dégrader à leur précision. A titre d'exemple, supposons que l'opération de rafraîchissement de l'exponentielle nécessite un cycle d'horloge et que $P_0$ vaut 100 µs, il est donc possible de rafraîchir 1000 exponentielles en multiplexant dans le temps l'opération d'un seul circuit.

De point de vue pratique, le multiplexage temporel consiste à placer une mémoire RAM dans le circuit en question pour alterner les valeurs intermédiaires aux ressources de calcul communes. A chaque créneau de temps, le circuit scrute une entrée de la mémoire, extrait l'opérande, réalise le calcul et réinsère la nouvelle valeur dans la mémoire. Le circuit multiplexé dédié au calcul de l'exponentielle est schématisé dans la figure 3.7.

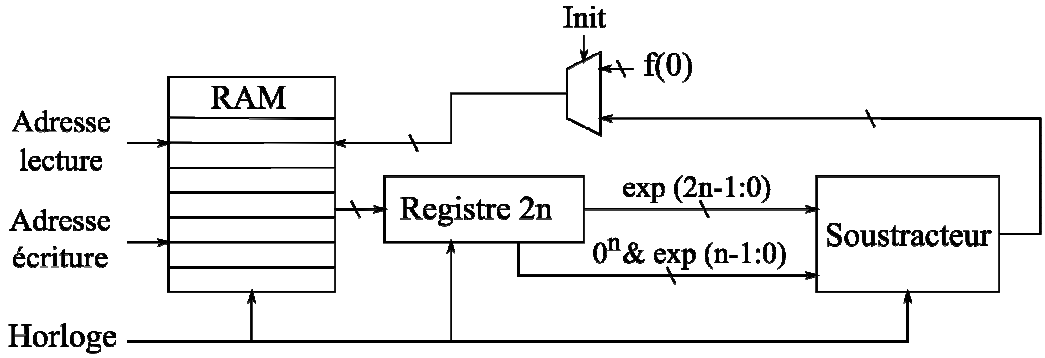



Figure 3.7. Circuit du multiplexage du calcul des exponentielles.

L'entrée *Adresse_écriture* représente l'adresse de l'entrée de la RAM qui sera utilisée pour insérer un opérande en case mémoire. L'opérande peut être la fonction *f(0)* ou le résultat issu du soustracteur. L'ordonnanceur gère l'attribution de ces entrées. Il spécifie également la valeur de l'entrée *adresse_lecture* qui détermine l'emplacement du prochain opérande qui sera flashée dans le registre 2n pour démarrer une opération de calcul.

Cette approche permet de réduire considérablement le besoin en ressources matérielles. Elle met en œuvre un seul circuit STDP pour le calcul de toutes les connexions plastiques de réseau. Elle réalise également un équilibre entre l'utilisation des mémoires distribuées et des blocs RAM. La précision de calcul est la même que celle obtenue par l'approche parallèle.

Cependant, l'inconvénient ici est le même dans chaque calcul séquentiel : un temps de traitement plus grand. En effet, le temps d'attente dans la FIFO additionné au temps d'accès aux blocs mémoire ralentit la latence du système. Cette réalisation reste tout de même plus efficace qu'une réalisation à l'aide d'un processeur embarqué parce qu'elle est spécifique au calcul et réalise des optimisations au niveau de l'implémentation des composants de calcul.

Une quantification du temps de traitement en fonction du nombre de connexions plastiques est nécessaire pour conclure sur la fiabilité de cette réalisation vis-à-vis des contraintes de fonctionnement relatives au système PAX2. Cette approche a été concrétisée par une implémentation sur FPGA. Une évaluation de ses performances est donnée dans la section suivante.

Après la réalisation de l'approche parallèle et l'approche séquentielle, la question se pose sur une approche mixte qui tire profit des avantages des deux premières : un temps de traitement réduit et une économie dans l'utilisation des ressources du FPGA. Le paragraphe suivant présente une autre façon de voir les choses, ce qui rendra possible la combinaison des deux approches.

*Calcul de la plasticité du réseau : approche orientée-cellule*

Les deux approches présentées précédemment reposent sur la notion de connexion pour calculer la plasticité. Ceci revient à dire que pour réaliser la plasticité d'un réseau de $N$ neurones, on a besoin de couvrir les $N^2$ connexions qui le constituent. Or, une connexion peut être vue aussi comme un couple de neurones pré et postsynaptique qui, selon leur activité, modulent une variable intermédiaire, le poids synaptique.

Le calcul de la plasticité peut être vu comme un traitement pré et postsynaptique indépendant dont le résultat est utilisé pour déterminer la nouvelle valeur du poids. Cette vision du problème, sépare le calcul en deux parties : une partie qui se fait au niveau de chaque neurone et une deuxième partie qui nécessite la contribution d'un couple de neurone.

L'approche *orientée-cellule* consiste à séparer les deux parties : les calculs relatifs à l'activité d'un neurone membre d'une connexion sont réalisés indépendamment des autres activités des neurones. A l'arrivée d'une séquence particulière d'évènements pré et postsynaptiques, les résultats des calculs relatifs aux cellules seront exploités et combinés dans un dernier calcul qui combine, cette fois ci, les contributions du couple de neurones.

Etant donné que le nombre de neurones n'est pas grand (25), le calcul relatif à l'activité d'un neurone peut être parallélisée. Les problèmes liés à la limitation des ressources matérielles rencontrés lors du développement de l'approche parallèle, sont contournés dans cette approche puisque l'on n'a plus affaire à des connexions à implémenter ($N^2$) mais plutôt à des neurones ($N$). La deuxième partie qui réalise le calcul commun à un couple de neurones suit un traitement séquentiel, puisqu'elle doit couvrir tous les couples de neurones dans le réseau ($N^2$). En résumé, l'approche orientée-cellule ouvre la voie à une implémentation mixte parallèle-séquentielle en relativisant le calcul de la plasticité aux cellules.

De point de vue pratique, l'algorithme de plasticité utilise les évènements d'un neurone pour calculer les fonctions P, Q, $\varepsilon_j$ et $\varepsilon_i$. Ces fonctions peuvent donc être ″proches″ du neurone. Autrement dit, un circuit permettant de calculer ces fonctions est attribué à chaque neurone. Le circuit est illustré par la figure 3.8.

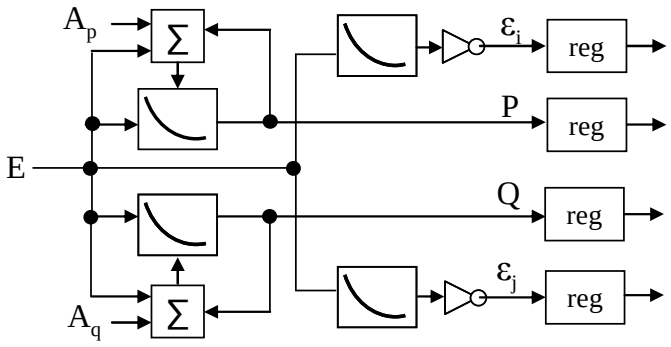



Figure 3.8. Schéma bloc du circuit du calcul des fonctions de plasticité du côté d'un neurone.

L'entrée E achemine les évènements du neurone aux blocs de calcul. Les exponentielles et les circuits des fonctions de plasticité sont calculés de la façon décrite dans le paragraphe 3.1. Le résultat de chaque fonction est sauvegardé en permanence dans un registre accessible de l'extérieur. Une lecture des registres permet d'extraire les valeurs courantes des fonctions.

Le calcul d'un nouveau poids d'une connexion met en œuvre 2 circuits, un pour le neurone présynaptique et un deuxième pour le neurone postsynaptique. L'ordre d'arrivée des évènements aux entrées E sélectionne le type de l'opération plastique à réaliser (LTP ou LTD). Dans le cas d'une LTP, les fonctions P et $\varepsilon_j$ seront lues à partir des registres du neurone présynaptique et la fonction $\varepsilon_i$ à partir des registres du neurone postsynaptique. Et inversement, lorsqu'on a une LTD, les fonctions Q et $\varepsilon_i$ seront lues à partir des registres du neurone postsynaptique et la fonction $\varepsilon_j$ à partir du registre du neurone présynaptique.

La détermination des connexions se fait à l'aide d'un ordonnanceur qui scrute les données de configuration du réseau d'une façon séquentielle. L'ordonnanceur réagit à des évènements stockés dans la FIFO des PAs (Potentiels d'Action). Il envoie alors une requête constituée des coordonnées des neurones pré et postsynaptiques au bloc de *calcul de poids*. Ce dernier réalise le calcul de la variation du poids en scrutant les valeurs des fonctions STDP relatives aux neurones formant la connexion. Il implémente toute la série de multiplications et additions/soustractions nécessaires pour le calcul. La figure 3.9 montre le schéma bloc correspondant à la réalisation globale selon l'approche orientée-cellule.

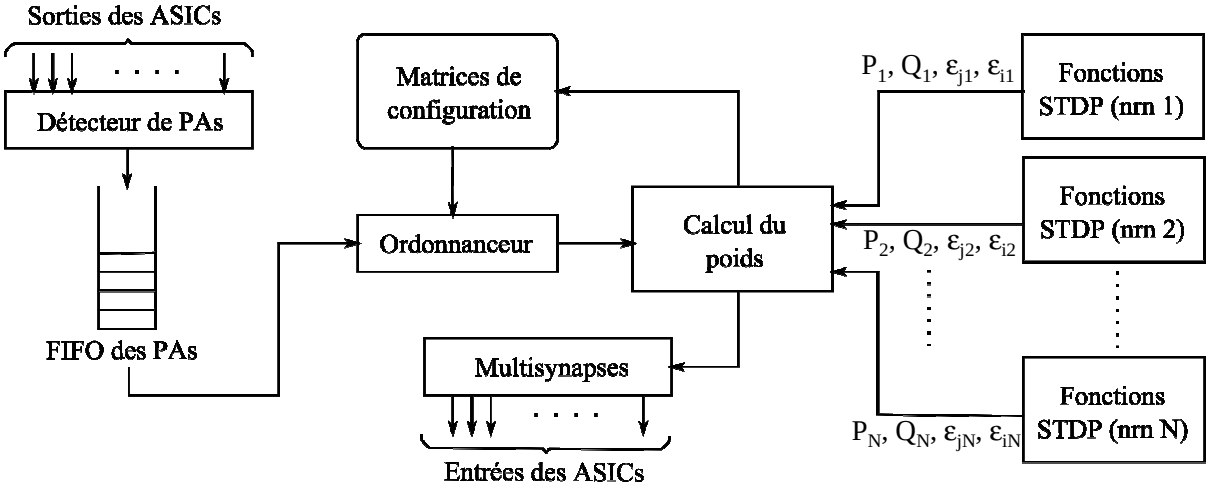



Figure 3.9. Schéma bloc du circuit d'une réalisation de l'approche orientée-cellule.

La parade de cette approche est la séparation des composants gourmands en termes de ressources matérielles (additionneurs et multiplieurs) des autres composants (calcul des exponentielles). Ces derniers ont pu être parallélisés pour apporter plus de performances au calcul de la plasticité.

La différence majeure entre cette réalisation et la réalisation purement séquentielle vient du fait que les circuits du calcul permanent ne subissent pas une sérialisation. Ils sont dupliqués autant de fois qu'il y a de neurones dans le réseau. Par conséquent, les résultats des calculs permanents sont immédiatement disponibles en sortie et ne nécessitent pas de temps et de traitement supplémentaires pour les extraire, comme c'est le cas dans la réalisation séquentielle.

### 3.2 Comparaison des performances

Les trois réalisations présentées dans la section précédente ont été concrétisées par trois implémentations différentes sur un même FPGA. Après placement et routage, l'utilisation des ressources matérielles correspondantes à chacune des implémentations est donnée dans le tableau 3.2.

La réalisation parallèle consomme tous les multiplieurs dédiés disponibles et utilise des LUT[1] supplémentaires pour en câbler d'autres. Il en résulte qu'une grande quantité de LUTs est utilisée, ce qui limite la fréquence maximale du FPGA à 66 MHz. Les autres réalisations ne présentent pas cet inconvénient et utilisent seulement 4 multiplicateurs dédiés. Ils supportent largement la fréquence du fonctionnement de le FPGA (100 MHz).

La réalisation séquentielle utilise des blocs RAM pour multiplexer temporellement ces circuits, tandis que la réalisation orientée-cellule utilise plus la mémoire distribuée sous forme de registres (Belhadj, 2009b). D'une façon générale, ces deux réalisations présentent une faible utilisation des ressources du FPGA par rapport au traitement qu'elles offrent.

Tableau 3.2. Allocation des ressources matérielles pour les trois réalisations du calcul de la plasticité.

| **Réalisation** | **Fréquence maximale** | **Bloc RAM** | **Multiplieur dédié** | **LUT (ou bloc logique)** |
|---|---|---|---|---|
| Parallèle | 66 MHz | 0 | 32 (100%) | 20205 (76%) |
| Séquentielle | 107 MHz | 5 (10%) | 4 (10%) | 1184 (4%) |
| Orientée-cellule | 113 MHz | 1 (3%) | 4 (10%) | 3079 (11%) |

L'utilisation des ressources constitue une analyse spatiale des performances des réalisations. Pour compléter la comparaison entre les réalisations, une analyse temporelle est nécessaire.

Le calcul de la plasticité est limité temporellement par une contrainte temps-réel, fixée à 10 ms. Le nombre de connexions qui peuvent être traitées dans cet intervalle de temps reflète la puissance de calcul de la réalisation. Le tableau 3.3 mesure la puissance de calcul relative

[1] La LUT, ou Look-Up-Table, est un composant dans le FPGA qui sert à implémenter des équations logiques.

aux trois réalisations, ainsi que le nombre maximal de connexions que peut atteindre chaque réalisation.

Tableau 3.3. Les performances du calcul de la plasticité.

| **Réalisation** | **Puissance de calcul (connexions/s)** | **Nombre maximal de connexions** | **Nombre maximal de neurones** |
|---|---|---|---|
| Parallèle | 200 M | 132 | 11 |
| Séquentielle | 4 M | 20 K | 141 |
| Orientée-cellule | 10 M | 50 K | 223 |

La réalisation parallèle présente la plus grande puissance de calcul avec 200 M connexions/s mais seulement pour 132 connexions. Ce nombre modeste de connexions reflète la limitation du matériel vis-à-vis de l'approche parallèle. La taille maximale des réseaux que l'on peut simuler ne dépasse pas les 11 neurones ($\sqrt{132}$), ce qui n'est pas suffisant pour gérer la plasticité dans le système PAX2.

La réalisation séquentielle peut effectuer jusqu'à 4 M connexions/s permettant la simulation de réseaux de 141 neurones en temps réel. Ce nombre est largement suffisant pour le système PAX2. L'approche orientée-cellule qui représente un meilleur compromis parallèle-séquentiel est capable de traiter jusqu'à 10 M connexions/s en temps réel, soit 50 K connexions/10ms. La taille maximale du réseau est presque le double de celle obtenue par la réalisation séquentielle.

En résumé, une comparaison spatiale et temporelle des trois approches montre que seule l'approche parallèle ne peut pas être exploitée dans le système PAX2. Malgré le fait qu'elle possède de bonnes performances temporelles, elle souffre d'une utilisation accrue des ressources matérielles. Les deux autres approches peuvent être exploitées et utilisées pour le calcul de la plasticité. Toutes les deux satisfont les contraintes indiquées dans le cahier des charges. Cependant, plus les calculs sont rapides, plus la latence est faible et plus la réponse du système est précise. Grâce à son implémentation mixte parallèle-séquentielle, l'approche orientée-cellule possède le meilleur compromis puissance de calcul/ressources matérielles. Elle sera retenue pour le système PAX2.

### 3.3 Le fonctionnement global du système

Dans cette partie, on va rassembler tout les composants du système pour expliquer son fonctionnement global. Les explications sont fournies au niveau fonctionnel : les détails de la communication entre les composants ainsi que leurs limites physiques ne sont pas représentés.

Le fonctionnement global du système est illustré par la figure 3.10 qui détaille la boucle de simulation dans un réseau à 3 neurones dont la spécification est donnée dans la figure 3.1 (a). Le traitement est centralisé autour d'un module de calcul de la plasticité. Les sorties des neurones analogiques sont directement connectées aux entrées d'un encodeur. Ce dernier scrute périodiquement ses entrées afin de détecter un potentiel d'action. A l'arrivée d'un potentiel d'action, l'encodeur code en sortie l'adresse de l'entrée correspondante. Une heure d'apparition est associée au nouvel évènement qui sera par la suite sauvegardé dans une FIFO.

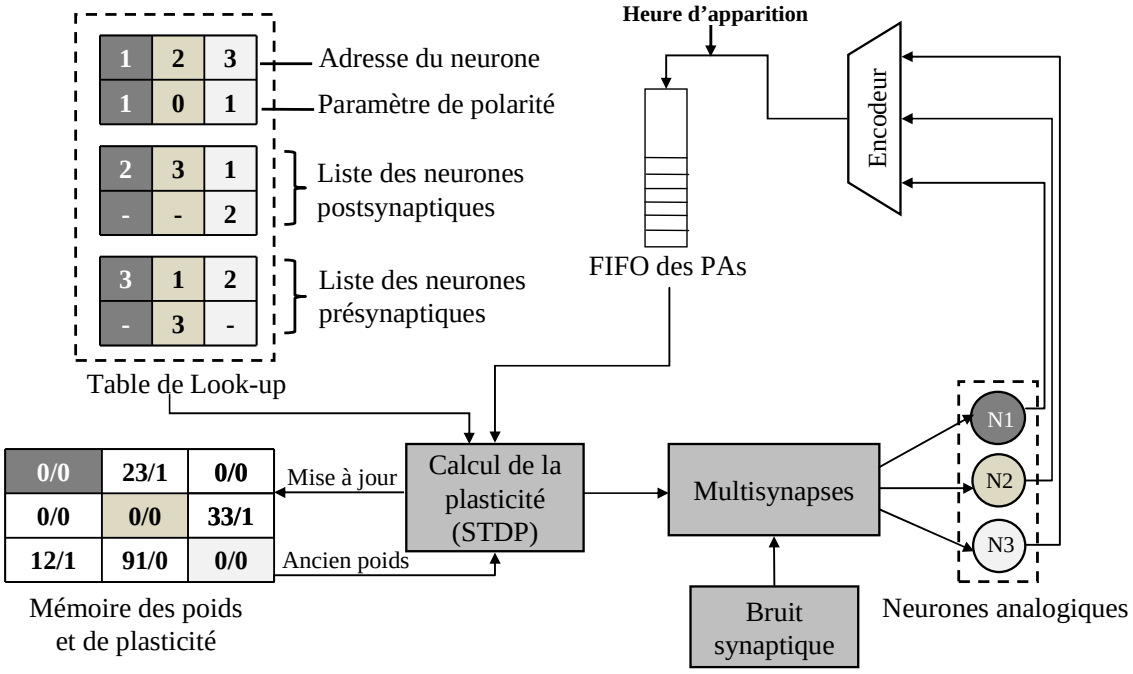



Figure 3.10. Illustration du fonctionnement global du système PAX2.

Le module de plasticité tire les évènements un par un de la FIFO et réalise les étapes suivantes pour chaque évènement :

- En utilisant l'adresse du neurone à l'origine du potentiel d'action, il envoie une requête à la table de Look-up pour chercher l'entrée correspondante au neurone. La table de Look-up est une mémoire RAM qui contient les informations de chaque neurone du réseau. Chaque colonne de la table contient les informations d'un neurone dont son adresse, sa polarité, la liste des neurones postsynaptiques et présynaptiques qui forment avec lui des connexions.
- Le module de plasticité scrute alors les listes des neurones pré et postsynaptiques pour construire les coordonnées des connexions. Les coordonnées sont composées par le couple (*pre,post*) dont le premier terme est l'adresse du neurone à l'origine de l'évènement et le second l'adresse du neurone de la connexion synaptique concernée. Ces coordonnées permettent de chercher les informations relatives à chaque connexion dans la mémoire des poids et de plasticité. Cette mémoire représente les matrices W et P de l'étape de configuration.
- Les poids synaptiques issus d'une connexion dont le neurone postsynaptique n'est pas le neurone à l'origine du potentiel d'action sont directement envoyés au module de la multisynapse pour activer la synapse correspondante.
- L'étape suivante consiste à appliquer les règles de la STDP en utilisant le poids synaptique et les heures d'apparition des évènements pré et postsynaptiques. Dans le cas où la connexion est plastique le calcul est directement déclenché pour donner à la fin une nouvelle valeur du poids. Ce dernier remplacera l'ancien dans la mémoire des poids et de plasticité par une opération de mise à jour. Cette étape est répétée jusqu'à la fin des listes des neurones pré et postsynaptiques.

Ensuite, les mêmes étapes se répètent pour chaque évènement de la FIFO. En parallèle, chaque évènement généré est considéré comme un résultat de simulation et est envoyé à la machine hôte, associé à son heure d'apparition. De même, chaque nouveau poids synaptique calculé est transmis vers l'ordinateur, avec les coordonnées de la connexion qui la représente. Les résultats ainsi récupérés serviront pour une analyse post-simulation afin de pouvoir interpréter les dynamiques d'apprentissage et d'adaptation du réseau.

Les multisynapses transforment les poids synaptiques en impulsions qui activent les synapses des neurones postsynaptiques. Ces derniers peuvent émettre des potentiels d'action fermant ainsi la boucle de simulation.

Le module du bruit synaptique agit directement sur les multisynapses de tous les neurones. Il est programmé de telle sorte qu'il fournit des poids synaptiques à valeur constante à des intervalles de temps prédéterminés (ISI) selon une distribution de Poisson (Daouzli, 2009). Ces poids sont additionnés à ceux qui sont relatifs aux connexions pour contribuer à l'activation des entrées synaptiques.

Durant une simulation, le bruit synaptique stimule en permanence les entrées synaptiques. Il assure une certaine activité de fond pour le réseau. Sans bruit synaptique, l'activité de tout le réseau risque de s'arrêter et ne plus produire d'évènements à cause des phénomènes reliés à l'auto-stabilisation des réseaux de neurones.

## 4 Mise en réseau et validation

La mise en réseau est la phase finale de la réalisation du système qui permet d'exploiter la plateforme ainsi développée. Elle comprend, entre autres, la communication avec la machine hôte, la récupération des résultats de simulations et l'interfaçage des composants du système.

### 4.1 La récupération des résultats de simulation

Les résultats de simulation reflètent l'évolution de la plasticité du réseau ainsi que l'activité des neurones. A la fin d'une simulation, on souhaite tracer les activités des neurones et l'évolution des poids synaptiques qui régissent les connexions plastiques.

La carte Gaillimh ne peut être reliée à la machine hôte que par des liaisons série RS232. Trois liaisons ont donc été installées. La première est utilisée pour envoyer la configuration vers les composants du système et rafraîchir le module du bruit synaptique en cours de simulation (voir paragraphe suivant). Elle fonctionne à un débit de 38400 bits/s. Les deux autres liaisons sont utilisées alternativement pour la récupération des résultats de simulation avec un débit côtoyant les 2 Mbits/s. L'annexe A détaille le fonctionnement des 3 liaisons.

Si on se positionne dans les conditions du pire cas, le débit de 2 Mbits/s ne permet pas de récupérer les résultats de simulation que pour un réseau constitué de 13 neurones (réseau totalement connecté). Il est possible, cependant, d'élargir la taille de réseau en réduisant sa connectivité (réseau partiellement connecté) ou la fréquence de génération des potentiels d'action. Cette nouvelle limite contraint l'utilisateur final en imposant des restrictions d'utilisation de la plateforme, ce qui contredit la flexibilité de configuration initialement indiquée dans le cahier des charges. On a choisi de limiter le nombre maximal de neurones utilisé dans les simulations à 13 neurones (parmi les 25 disponibles), pour garder la possibilité de configurer des réseaux *all-to-all*. Malgré le fait que les réseaux complètement connectés sont rares en biologie, ils sont tout de même utilisés dans nos simulations.

### 4.2 La stimulation et le bruit synaptique

L'activation des synapses consiste à coder la valeur d'un poids synaptique en une impulsion. Cette dernière modifie, positivement ou négativement, le potentiel de membrane. Du côté de la partie numérique du système, cette opération est assurée par le module de la multisynapse. Ce module somme les valeurs des poids synaptiques incidents et génère l'impulsion. Pour un neurone analogique, une stimulation maximale correspond à une impulsion de 60 µs de largeur (une contrainte analogique des ASICs). Il en résulte que la somme des poids incidents à une multisynapse ne doit pas dépasser une certaine valeur maximale qui conduit à une impulsion plus large que 60 µs.

Pour le système PAX2, on utilise au maximum 13 neurones. La plus grande somme des poids dans une multisynapse est le résultat d'une activité simultanée de tous les neurones présynaptiques faisant intervenir des valeurs maximales du poids. Au total, on peut avoir une somme maximale égale à 13*poids maximal. Pour un poids codé sur 8 bits, la somme maximale est de 3315 unités poids.

En allouant l'équivalent d'un cycle d'horloge (10 ns) à chaque unité poids, on aura une impulsion de largeur maximale 33,15 µs. Cette valeur reste inférieure à 60 µs et ne pose pas de problème de dépassement. Cependant, cette valeur risque d'augmenter avec les stimulations envoyées par le bruit synaptique.

Le bruit synaptique additionne une valeur maximale d'un poids (255) à la somme d'une multisynapse à la fin de chaque ISI. Les ISIs sont de l'ordre d'une centaine de millisecondes, présentant ainsi une échelle temporelle plus grande que celle de l'impulsion de stimulation. Au pire des cas, la largeur d'une impulsion maximale se prolonge de 2,55 µs.

Les valeurs des ISIs sont calculées par une application logicielle. Ils sont par la suite envoyés à des emplacements mémoires spécifiques dans le FPGA, où ils seront interprétés par des modules spécifiques qui comptent l'intervalle de temps indiqué avant d'envoyer une requête de stimulation à la multisynapse. Pour conserver les ressources mémoire dans le FPGA, ces mêmes emplacements seront rafraîchis périodiquement par la machine hôte avec des valeurs des nouveaux ISIs. Le protocole de communication qui permet de réaliser cette opération est décrit en détail dans l'annexe A.

### 4.3 La détection des potentiels d'action

Le détecteur de potentiels d'action est une machine d'état qui scrute les sorties des comparateurs des ASICs. Le comparateur sort un 1 logique si il y a un potentiel d'action, 0 sinon. Un potentiel d'action (1 logique) dure entre 1 et 5 millisecondes, selon la dynamique des neurones. La détection d'un potentiel d'action revient à détecter le passage de 0 à 1 logique de la sortie du comparateur. Cependant, des subtilités de fonctionnement font que le signal sortant du comparateur réalise des rebonds successifs non seulement lors du passage du 0 à 1, mais aussi lors du passage de 1 à 0, comme le montre la figure 3.11.

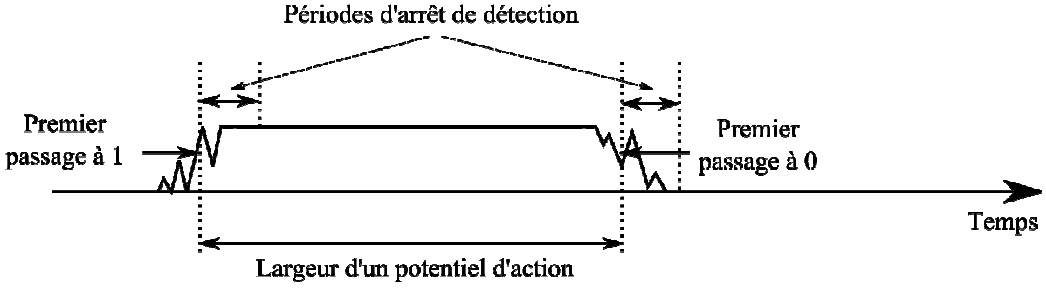



Figure 3.11. La forme du signal sortant du comparateur.

Pour résoudre ce problème, un filtre anti-rebond classique est ajouté en amont de la machine d'état de détection des potentiels d'action. Ce filtre consiste à ignorer complètement les variations du signal pendant une période de 200 µs, aussi appelée période d'arrêt de détection. L'action du filtre commence au premier passage à 1 logique ainsi qu'au premier passage à 0 du signal à la sortie du comparateur. Il permet ainsi d'éviter la détection de plusieurs potentiels d'action au lieu d'un seul.

### 4.4 Les outils de mise à disposition du système

Deux outils logiciels ont été développés pour le système PAX2. Le premier outil sert à générer les paramètres de configuration d'une simulation à partir d'une spécification du réseau de neurones. La spécification est réalisée par l'intermédiaire d'un package Python[1], appelé PyNN. Actuellement, PyNN supporte plusieurs plateformes de simulation de réseau de neurones, logicielles et matérielles. A. Daouzli a ajouté un module contenant les spécificités de la plateforme de simulation PAX2 à ce package durant les travaux de sa thèse, permettant ainsi une spécification standardisée des simulations (Daouzli, 2009).

L'application de l'outil PyNN pour le système PAX2 génère un fichier contenant tous les paramètres nécessaires à une simulation. Ces derniers sont donc envoyés à la carte *Gaillimh*. Le début et la durée de la simulation sont spécifiés en avance par l'utilisateur final. Lorsque la simulation commence, un deuxième outil, appelé *gail_sim*, est utilisé pour récupérer les résultats de simulation. Cet outil, également développé par A. Daouzli, génère en sortie l'évolution de la fréquence moyenne d'oscillation de chaque neurone, le *raster plot*[2] des activités des neurones et l'évolution des poids synaptique des connexions en fonction du temps.

### 4.5 Des exemples de simulation

Avant de commencer une simulation, une phase de réglage des circuits analogiques est nécessaire pour spécifier le type du neurone. Cette phase est particulièrement difficile à mettre en œuvre. En effet, chaque neurone nécessite un réglage spécifique même s'il simule un même type que d'autres neurones.

Pour une première simulation, nous avons réglé 4 neurones, *N1*, *N2*, *N3* et *N4*, configurés en RS (*Regular Spiking*) avec des fréquences moyennes d'oscillation de 70, 50, 30 et 10 Hz respectivement. Les poids synaptiques ont été initialisés avec une valeur de 120. Les paramètres de plasticité sont ceux du modèle de plasticité original, soient $\tau_q$=33.8ms, $\tau_p$=14.8ms, $\tau_i$=88ms, $\tau_j$=28ms, $A_p$=0.1, $A_q$=0.05, $w_{LTP}$ = 255 et $w_{LTD}$ = 0.

Les 4 neurones sont interconnectés comme le montre la figure 3.12 (a) : une connectivité mutuelle entre deux neurones voisins. Avec les fréquences d'oscillations des neurones, mentionnées plus haut, cette configuration stipule que, dans une direction, les connexions subiront une potentiation, alors que dans la direction inverse, elles subiront une dépression. Ceci vient du fait que l'un des deux neurones voisins présente une fréquence intrinsèque plus rapide que le deuxième, ce qui fait qu'il y aura plus d'occasion de réaliser une LTP dans une direction, et de réaliser une LTD dans la direction opposée.

[1] Python est un langage de programmation interprété multi-paradigmes.

[2] Le *raster plot* est un type de graphe utilisé dans les neurosciences computationnelles pour tracer l'activité d'un ensemble de neurones en fonction du temps, en visualisant uniquement les temps d'apparition des potentiels d'actions.

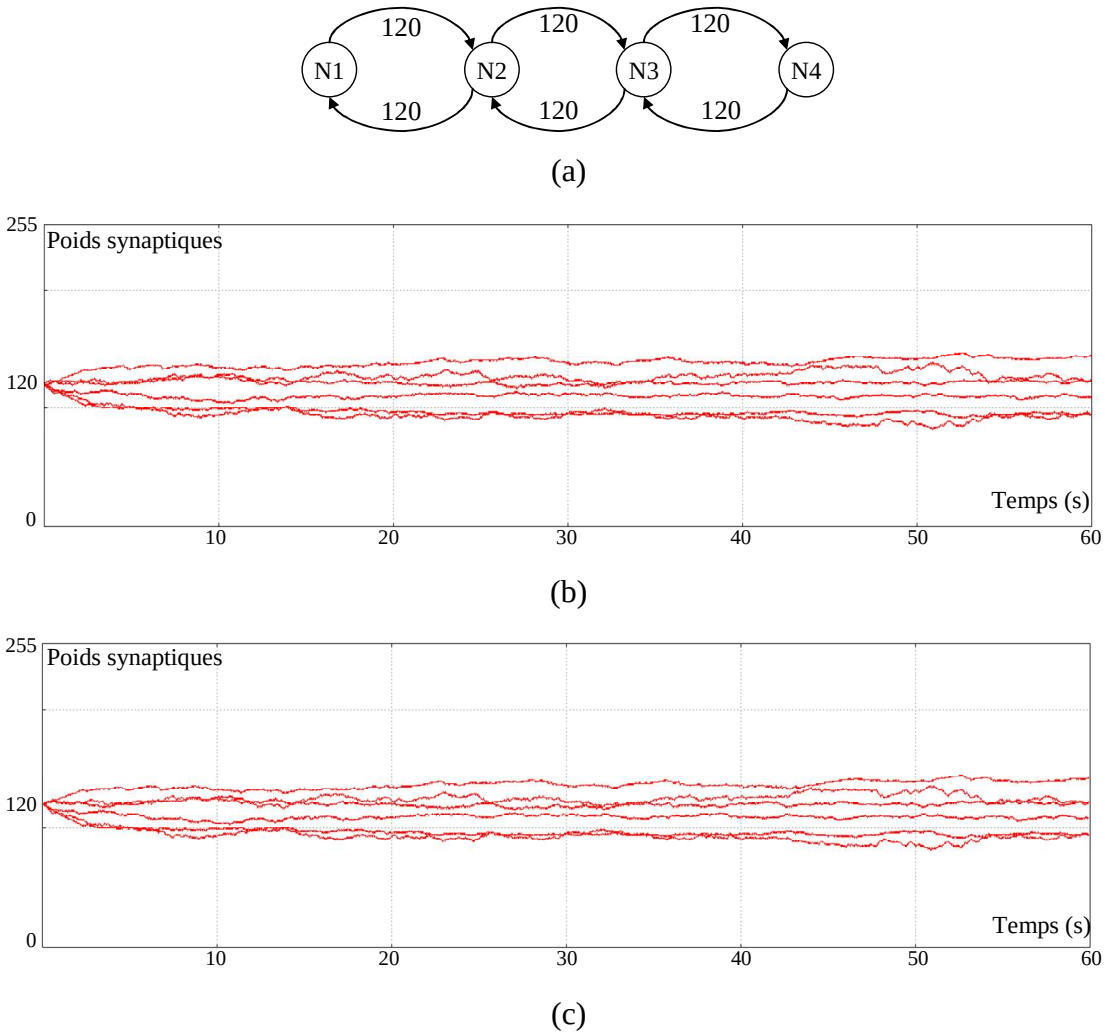



Figure 3.12. Comparaison de l'évolution des poids dans une simulation d'un réseau à 4 neurones. (a) La spécification du réseau de neurones. (b) Résultat de la simulation matérielle (virgule fixe). (c) Résultat de la simulation logicielle (virgule flottante).

En résumé, il est attendu que l'on aura 3 potentiations et 3 dépressions dans les évolutions des poids synaptiques. Vue la différence des fréquences d'oscillation, les évolutions ne seront pas confondues, il aura certainement 6 évolutions différentes. La figure 3.12 (b) représente les courbes d'évolutions des poids synaptiques durant une minute de simulation sur le système PAX2 (équivalent à une minute de temps réel biologique).

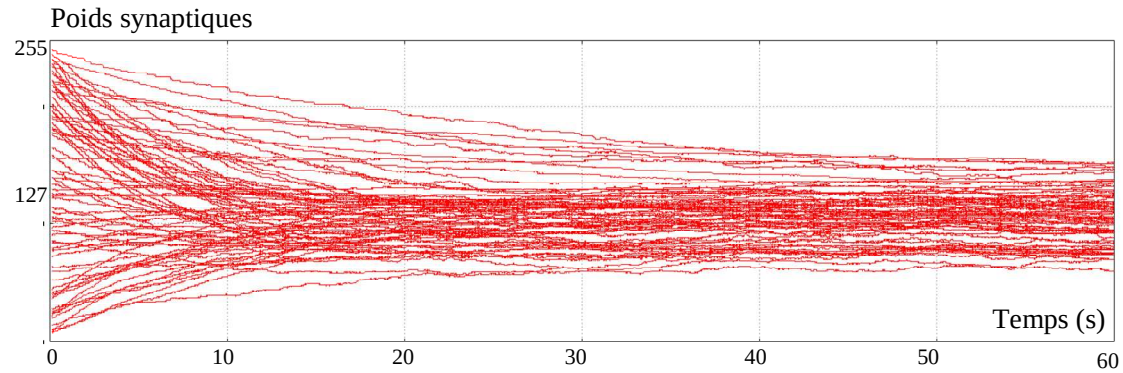
L'activité du réseau tend à se stabiliser spontanément. Sous l'effet stabilisant de la plasticité, les fréquences d'oscillation tendent vers une fréquence moyenne commune. La variation des poids synaptiques diminue en avançant dans la simulation. A la fin de la simulation, toutes les fréquences sont aux alentours de 48 Hz et les activités des neurones se synchronisent de plus en plus, ce qui explique la stagnation de l'évolution des poids à la fin de la simulation dans la figure 3.12 (b).

Pour vérifier le calcul de la plasticité du réseau, on a développé une version logicielle qui effectue le même calcul mais en virgule flottante (calcul plus précis). L'outil prend en entrée l'activité des neurones (potentiels d'action et heures d'apparition) recueillies de la simulation matérielle et itère avec un programme C l'algorithme de la STDP. Le résultat du calcul logiciel est donné dans la figure 3.12 (c). On observe que les comportements logiciels et matériels des poids synaptiques sont très proches.

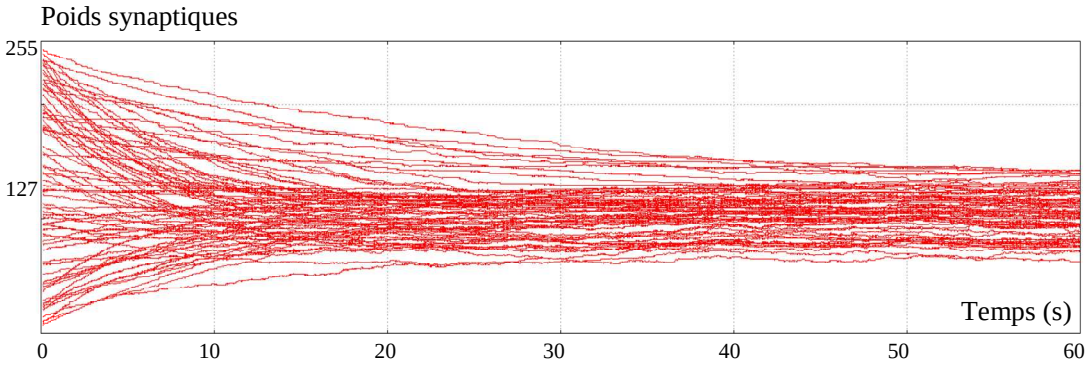
Dans une deuxième simulation, on cherche à accentuer la propriété d'auto-stabilisation inhérente au système PAX2. Un nombre de 9 neurones, tous de type RS (neurones excitateurs), ont été configurés. Le réseau est totalement connecté (*all-to-all*). Tous les neurones sont réglés pour osciller avec des fréquences constantes distribuées aléatoirement entre 6 et 21 Hz. Les poids initiaux sont également aléatoirement choisis dans une plage allant de 0 à 255. Les paramètres de plasticité sont les mêmes que dans la simulation précédente.

La figure 3.13 montre les résultats de l'évolution des poids obtenus à partir d'une simulation matérielle et logicielle. Les poids synaptiques partent de valeurs différentes pour tendre à la fin vers un état stable intermédiaire. Ce comportement est expliqué par l'effet régulateur de l'algorithme de plasticité. En effet, les poids synaptiques sont calculés en fonction de l'activité des neurones. Ils codent dans leurs variations les séquences des évènements selon les règles de plasticité. Ensuite, ils interviennent dans la stimulation des neurones où ils appliquent les nouveaux changements dans la boucle de simulation, influant ainsi les fréquences des neurones. L'itération de ces opérations fait en sorte que l'algorithme de plasticité est prépondérant dans l'évolution de la simulation vers un état final.

L'algorithme de plasticité que l'on utilise tend toujours à converger vers un état stable (Destexhe, 2004). Ce fait est illustré par la figure 3.13 où toutes les connexions convergent vers des valeurs intermédiaires, indépendamment des valeurs initiales des poids. Après une minute de simulation, les fréquences d'oscillation des neurones se stabilisent aussi autour de 19.8 Hz.




(a)




(b)

Figure 3.13. Mise en évidence du la propriété de l'auto-stabilisation dans le système PAX2. (a) Résultat de simulation matérielle (calcul à virgule fixe). (b) Résultat de simulation logicielle (calcul à virgule flottante).

La légère différence qui existe entre les points de convergence finaux des simulations logicielle et matérielle ne permet pas d'évaluer la précision du calcul de la STDP de la réalisation matérielle. En effet, il n'est pas possible de comparer des changements synaptiques utilisés dans une simulation réelle (avec rétroactions directes) à une simulation qui utilise le résultat d'une autre pour réaliser ses calculs. Autrement dit, si les calculs à virgule flottante remplacent les calculs à virgule fixe, l'activité des neurones sera aussi différente de celle obtenue par le calcul matériel. Un calcul logiciel sert principalement à vérifier le comportement général des poids synaptiques et non pas la précision de calcul.

## 5. Conclusion

Ce système prouve la faisabilité et la pertinence de la réalisation purement matérielle d'un simulateur de réseaux de neurones. La boucle de simulation est réduite à des accès entre les composants d'une même carte, sans faire appel à des traitements logiciels. Les simulations sont effectuées en temps réel pour un nombre plus grand de neurones.

La réussite d'une gestion de réseau en matériel est surtout le fruit d'une réalisation optimisée du calcul de la plasticité. Ce dernier constitue le cœur du calcul numérique du système, par lequel passe toutes les connexions du réseau. L'évolution quadratique du nombre de connexions en fonction du nombre de neurones nous oblige à multiplexer les blocs de calcul pour réduire l'utilisation des ressources. La réalisation qui permet d'atteindre la meilleure puissance de calcul est celle qui possède le plus petit retard dû au multiplexage.

Ce système permet en fin du compte de simuler des réseaux de taille maximale 13 neurones. Il est cependant nécessaire de concevoir une structure plus adaptée pour des systèmes capables de gérer un nombre de neurones plus important. Etant donné que la gestion matérielle du réseau de neurones est réalisable avec une seule carte, il est possible d'envisager un simulateur matériel ayant une structure multicarte.

# 4. PAX3 : un simulateur multicarte temps réel

Les versions du simulateur PAX présentées dans les chapitres précédents sont munies d'une gestion centralisée du réseau de neurones. Ce type de gestion souffre de plusieurs limitations. D'une part, les ressources matérielles des composants de calcul, d'une autre part, des goulets d'étranglement fonctionnels importants. Une architecture distribuée semble être la solution pour remédier à ces limitations. En effet, multiplier les ressources matérielles et distribuer le contrôle des unités de calcul permet une meilleure intégration des fonctions neuromorphiques. En contrepartie, les algorithmes de calcul et de contrôle doivent être écris en version distribuée. De plus, un support de communication supplémentaire est nécessaire pour rassembler les différentes sous-unités.

Ce chapitre décrit les extensions apportées au simulateur PAX dans le but d'élargir la taille du réseau. Un système matériel multicarte est mis en place. Chaque carte représente un sous-réseau de 20 neurones et est connectée à un bus de communication qui la relie aux autres cartes. La section 1 liste les principaux objectifs du cahier des charges du système PAX3. La section 2 présente le support de communication sur lequel repose tous les échanges entre les cartes. Un protocole de communication temps réel est défini pour assurer cette tâche. La section 3 décrit les modifications apportées à l'algorithme de plasticité pour l'obtention d'une version distribuée. Des techniques et des méthodes d'implémentation de la plasticité sont également exposées dans cette section. La dernière section aborde les problèmes de mise en réseau des sous-unités de calcul et montre des résultats obtenus à partir du système matériel.

## 1 Le cahier des charges

On se propose dans cette version d'offrir à l'utilisateur la possibilité de simuler jusqu'à 400 neurones en mode *all-to-all*. L'objectif dans ce chapitre n'est cependant pas de réaliser une intégration à grande échelle, mais d'appréhender et de quantifier les besoins en communication et en puissance de calcul pour un tel réseau.

### 1.1 La communication temps-réel

La communication dans un système multicarte est caractérisée par des échanges intenses de données entre les unités de calcul situées sur des cartes différentes. Souvent, les échanges passent par un canal de communication partagé reliant les cartes entre elles. Chaque échange fait appel à des procédures de multiplexage/démultiplexage des données, de routage vers l'unité cible, d'encapsulation/décapsulation des trames, de temporisation, etc. Le temps de

traitement de ces procédures induit des retards supplémentaires dans l'heure d'arrivée des informations utiles à leur(s) destination(s).

Chaque retard potentiel va de pair avec l'introduction de variations dans les dynamiques de calcul des fonctions neuronales dépendantes du temps. Ainsi, la réaction des canaux ioniques peut être altérée (ou erronée) suite à une stimulation synaptique tardive due à un retard dans l'heure d'arrivée d'un potentiel d'action. Le risque d'altération de l'état du potentiel de membrane est d'autant plus grand que le retard est grand.

De plus, les mécanismes d'apprentissage dépendants du temps (tels que la STDP) sont affectés par les retards de communication. L'algorithme de plasticité que l'on utilise prend en compte les heures d'apparition des potentiels d'actions pré et postsynaptiques pour calculer les changements de l'efficacité synaptique. Ici, le risque d'altération des résultats du calcul est double, puisque les deux instants des potentiels d'action peuvent subir des changements. Les changements synaptiques ainsi calculés vont être réutilisés par la suite dans d'autres calculs. Il faut donc veiller à ce que la simulation garde le niveau de réalisme souhaité en essayant de limiter l'impact de ces erreurs.

Une communication à travers un canal partagé ne peut se faire sans retards supplémentaires. L'erreur temporelle est désormais inévitable. On est donc obligé de quantifier le retard lors de la conception du système et de le borner dans une marge de valeurs la plus petite possible. Il faut également quantifier l'erreur de calcul et son impact sur le résultat final de simulation.

L'enjeu principal est de majorer le retard par une valeur limite et de garantir que l'heure d'arrivée des potentiels d'actions ne dépassera pas une heure limite quelque soit l'état du trafic sur le bus de communication. Dans la phase de conception du système PAX3, on a étudié l'impact des retards sur le processus de stimulation et l'algorithme de plasticité. Le but était de déterminer la valeur limite du retard que l'on peut tolérer. La cinétique de la procédure de stimulation et du calcul des fonctions de plasticité est de l'ordre de la milliseconde. Un retard de déclenchement de quelques microsecondes n'induit pas une grande différence dans le résultat final et l'erreur peut être négligée. On a choisi la valeur de 25 µs comme retard maximal de l'arrivée d'un évènement à sa destination finale. Cette valeur correspond au mieux au niveau de réalisme des fonctions neuronales que l'on veut implémenter.

En résumé, le retard induit par la communication partagée ne doit pas dépasser la valeur de 25µs. Cette déclaration constitue désormais la contrainte temps-réel pour la communication dans le système PAX3.

### 1.2 La plasticité pour 500 neurones

On garde le même algorithme de plasticité utilisé dans les versions précédentes. Par conséquent, les ressources et la précision de calcul requises pour le traitement d'une connexion plastique resteront les mêmes. Ce qui diffère dans cette version, c'est la manière selon laquelle la plasticité du réseau est calculée. En effet, la taille de réseau passe de 25 neurones localisés sur une seule carte (version PAX2) à 500[1] neurones distribués sur plusieurs cartes. Ce qui signifie que le nombre de connexions plastiques à gérer par carte passe de 625 à $10^4$ connexions.

La croissance du nombre de connexions à gérer par carte ne doit pas s'accompagner par un accroissement du temps de traitement des calculs synaptiques. La contrainte temporelle

[1] Le système ne peut contenir que 400 neurones. Le nombre de 500 neurones est utilisé dans la réalisation du module de calcul de la plasticité pour optimiser l'implémentation (utilisation des mémoires à 512 entrées).

discutée dans le cahier des charges de la version précédente PAX2 doit être respectée dans cette version également : le calcul des nouvelles valeurs des poids synaptiques est limité par le temps minimal entre deux potentiels d'action générés par un même neurone. Etant donné que la fréquence maximale d'oscillation des neurones est de 100 potentiels d'action/s, tous les poids synaptiques doivent être calculés et mis à jour dans les limites d'une période de 10 ms.

La nouveauté dans le calcul de la plasticité vient de la multitude des cartes dans le système. Chaque carte constitue un sous-réseau de 20 neurones. Deux types de connexions en découlent : les *connexions externes*, ou connexions qui relient deux neurones localisés sur deux cartes différentes, et les *connexions locales* qui représentent le lien entre les neurones localisés sur la même carte. La distribution des neurones sur plusieurs cartes nécessite un algorithme de plasticité distribué pour gérer toutes les connexions plastiques du réseau.

### 1.3 La configuration et l'adressage

La configuration des paramètres des ASICs, de la topologie du réseau et des poids synaptiques initiaux nécessite le formatage des données de configuration pour faciliter leur routage. Le numéro de la carte cible doit figurer dans l'entête de chaque trame. Chaque carte sera capable de sélectionner les trames qui lui sont adressées par l'identification de son numéro. Le système global peut connecter jusqu'à 20 cartes.

L'adressage d'un neurone d'une carte peut se faire de deux façons : adressage local ou adressage global. L'adressage local consiste à ajouter dans l'entête des trames les numéros de l'ASIC (2 bits) et du neurone dans l'ASIC (3 bits). En ajoutant le numéro de la carte (5 bits), ce mode d'adressage nécessite 10 bits par trame. Il est utilisé généralement dans la phase de configuration des paramètres des neurones et dans la phase de simulation pour faciliter quelques traitements locaux reliés au calcul de la plasticité. L'adressage global, quant à lui, alloue seulement 9 bits pour coder chaque neurone dans le réseau. Chaque neurone possède un numéro unique entre 0 et 511. L'ordre d'attribution des numéros aux neurones n'est pas important. Pour passer d'un mode d'adressage à un autre, une table de conversion associant les numéros globaux avec les numéros locaux permet d'identifier les neurones relatifs à chaque carte. Ce mode d'adressage ajoute une couche de transparence par rapport à l'utilisateur final. L'utilisateur n'a plus à se soucier des numéros des cartes et des ASICs ; il s'adresse directement aux neurones du réseau sans avoir besoin de connaître leurs positions physiques. Ce mode est également utilisé pendant la simulation et est très utile pour l'interprétation des résultats de simulation. Il permet aussi d'automatiser les étapes de génération des paramètres de configuration et de faire certaines optimisations.

L'annexe B décrit les formats des trames. Ces trames sont envoyées au système rack par l'intermédiaire d'une liaison USB. Les résultats de simulation sont récupérés en utilisant la même liaison. Une analyse post-simulation sera donc possible.

## 2. Le protocole de communication temps-réel

L'ingénierie neuromorphique a adopté la technique de multiplexage temporel pour aborder le problème d'une connectivité massive entre les sous-unités de calcul dans un système neuromorphique distribué. La raison de ce choix réside dans la différence en bande passante entre un neurone (une centaine de Hz) et un bus de communication numérique (quelques dizaines de MHz). Cette différence permet de remplacer des milliers de connexions point-à-point par quelques fils métalliques conducteurs et quelques milliers de transistors.

La technique de multiplexage temporel a été utilisée avec succès dans de nombreux domaines de la télécommunication (Schwartz, 1987) et l'informatique (Tanenbaum, 1989).

Les solutions existantes de mise en réseau emploient cette technique pour augmenter le nombre d'utilisateurs dans un réseau informatique ou de télécommunications. Dans les systèmes neuromorphiques, cette technique est plutôt utilisée pour augmenter le nombre de neurones qui interagissent entre eux.

Les solutions existantes de mise en réseau peuvent être reprises pour gérer les réseaux de neurones implémentés en matériel. Cependant, les concepteurs des systèmes et circuits neuromorphiques doivent s'adapter à la différence de l'ordre des grandeurs des deux types de réseaux.

Dans cette section, on présentera une tentative d'adaptation de la topologie d'accès anneau à jeton au système PAX3. Cette topologie d'accès est utilisée dans de nombreux réseaux informatiques existants, tels que *Token Ring* (Carlo, 1998) et FDDI[1] (Wendy, 1993). Elle permet d'organiser l'accès à un canal de communication partagé avec un contrôle temporel permanent. Des applications informatiques temps-réel ont été réalisées avec succès en utilisant cette topologie (Malcolm, 1993).

### 2.1 La topologie d'accès anneau à jeton

Plusieurs topologies d'accès ont été utilisées avec succès dans les réseaux informatiques classiques. Des techniques d'accès tels que l'arbitration (maître/esclave), ALOHA[2], CSMA[3] et l'accès par priorité ont été adaptées et implémentées dans des systèmes neuromorphiques multicartes/multicircuits (Culurciello, 2003). Le choix d'une technique ou d'une autre dépend de l'application cible. Les exigences d'une application se présentent généralement en termes de débit d'accès, de latence et de consommation d'énergie.

La référence (Culurciello, 2003) présente une étude comparative entre plusieurs topologies d'accès utilisées dans les systèmes neuromorphiques. Les auteurs quantifient les compromis rencontrés lors de l'allocation de la bande passante, de l'attribution du temps d'accès au canal partagé, du temps écoulé dans les files d'attente et du débit exigé.

A notre connaissance, la topologie d'accès anneau à jeton n'a pas encore été adaptée et utilisée dans les systèmes neuromorphiques multicartes/multicircuits. Dans les réseaux informatiques, elle fut mise au point pour surmonter les inconvénients liés à la méthode de transmission CSMA/CD[4] dans les réseaux informatiques. Son but est de faire de sorte que toutes les stations aient une chance égale d'émettre, même lorsque le trafic sur le réseau est intense.

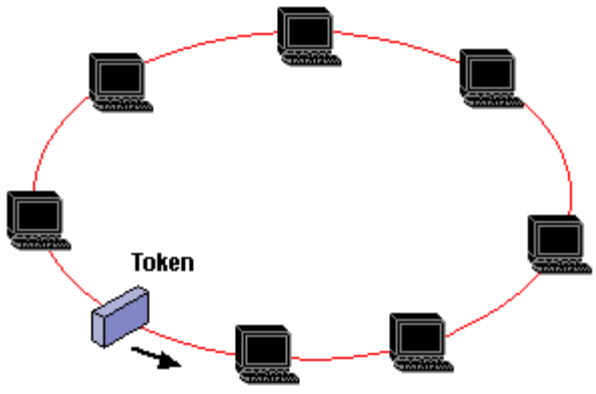



Figure 4.1. Architecture de la topologie d'accès anneau à jeton

[1] Fiber Distributed Data Interface.
[2] Un réseau de transmission de données faisant appel à un média unique.
[3] Carrier Sense Multiple Access.
[4] Carrier Sense Multiple Access with Collision Detection.

Le principe d'un protocole anneau à jeton est d'accorder à chaque station un créneau de temps prédéfini pour transmettre ses paquets. Cela se fait par la circulation d'un jeton autour d'un anneau (voir figure 4.1). Le jeton passe d'une station à une autre dans un ordre défini. Une station en possession du jeton peut décider de commencer une transmission, ou simplement de passer le jeton à sa voisine. A la fin du créneau de temps imparti, la station doit passer le jeton à la suivante, qu'elle ait ou non fini de transmettre ses paquets.

En plus de sa simplicité de mise en œuvre (pas de gestion de collisions), la topologie d'accès anneau à jeton garantit un certain niveau de performances, y compris la performance de transmission temps-réel. En effet, une transmission gouvernée par un créneau de temps prédéfini est une condition nécessaire pour une communication temps-réel. Cependant, cette méthode exige que toutes les stations du réseau participent correctement. Si tel n'était pas le cas, par exemple une station tombe en panne, c'est l'ensemble du système qui pourrait s'arrêter et qu'il faudrait réinitialiser.

L'architecture du système PAX3 a été conçue pour supporter une topologie d'accès anneau à jeton. Il faut cependant définir un algorithme d'accès pour spécifier les créneaux temporels relatifs à chaque nœud. L'algorithme d'accès permet d'allouer la bande passante du canal partagé d'une façon équitable entre les nœuds tout en respectant les contraintes temps-réel du système. Pour ce faire, on a choisi d'adapter à notre système une méthode de partage utilisée dans les réseaux FDDI (Malcolm, 1993). Dans son application originale, cette méthode a permis l'exécution en temps réel des applications distribuées sur plusieurs nœuds du réseau. Son objectif principal est de proposer des solutions avec une garantie de l'heure d'arrivée des messages à leur destination, dans des limites préfixées par l'utilisateur. La section suivante redéveloppe cette méthode en l'adaptant à notre problématique, à savoir la majoration du retard dû à la communication inter-cartes.

### 2.2 La méthode intégrative pour l'allocation de la bande passante

Cette section formalise le problème du retard de communication dans un système multicarte doté d'une topologie anneau à jeton. Dans un premier temps, on modélisera le réseau et les messages tels qu'ils sont aperçus dans un système neuromorphique multicarte. Dans un second temps, on fournira le support théorique nécessaire pour définir une méthode d'allocation équilibrée de la bande passante du canal partagé. La méthode ainsi définie permet de garantir un retard maximum dans l'heure d'arrivée des messages à leur destination. L'implémentation effective de cette méthode sur le système PAX3 fera l'objet de la section suivante (Belhadj, 2009a).

*Modèle du réseau*

Le réseau contient n nœuds arrangés en boucle autour d'un anneau à jeton. Le jeton transite d'un nœud *i* à un nœud *i+1* en gardant le même ordre de rotation. Le nœud qui possède le jeton transmet ses messages tandis que les autres nœuds se mettent à l'écoute. Chaque nœud accueille N neurones. Les neurones sont des générateurs d'évènements (potentiels d'action, instants des potentiels d'action, poids synaptiques,..). Les évènements seront encapsulés dans des messages qui seront transmis sur le canal de communication. Les messages qui attendent d'être transmis sont stockés dans des tampons selon la procédure FIFO. La transmission des messages se fait par diffusion (un-à-plusieurs).

Un réseau est caractérisé par les paramètres suivants :

- TTRT (*Token Target Rotation Time*) est le temps attendu pour que le jeton effectue un tour complet autour de l'anneau,

- $\tau$ est le temps de transition du jeton. Il accumule les temps de passage d'un jeton d'un nœud à un autre. Il représente la proportion de TTRT qui n'est pas utilisable pour la transmission des messages sur le canal de communication.

*Modèle des messages*

Il y a n flux de messages, $S_1$, … $S_n$, avec $S_i$ qui est le flux incident au nœud i. $S_i$ peut contenir deux catégories de messages : les messages à contrainte temps réel (*real-time constrained message*) et les messages à contrainte espace mémoire (*memory constrained message*). Les messages à contrainte temps réel concernent les instants d'émission des potentiels d'action générés par les neurones, tandis que les messages à contrainte espace mémoire sont les résultats des changements synaptiques au cours du temps. Chaque message à contrainte temps réel, ou potentiel d'action, doit être délivré au nœud cible avant l'expiration d'une heure limite, qu'on nommera ici D. Cette contrainte impose une limite exigée par l'utilisateur pour éviter les erreurs dues au retard de transmission sur les canaux partagés. Au contraire, les messages à contrainte espace mémoire n'ont pas de limite sur l'heure d'arrivée à la destination. Leur seule contrainte est de devoir être émis avant que la taille de la mémoire tampon soit atteinte et cause un problème de débordement de mémoire.

Chaque flux de messages $S_i$ est caractérisé par les paramètres suivants :

- $A_i$ représente l'activité d'un nœud i. Ce paramètre reflète le taux de génération de potentiels d'action par le nœud i. Par conséquent, il reflète le flux des messages à contraintes temps réel du nœud i.
- $C_i$ représente la connectivité du nœud i. Ce paramètre sert à prédire le nombre des changements synaptiques qui vont avoir lieu suite à la génération d'un potentiel d'action donné. Connaissant la valeur de ce paramètre ainsi que la valeur du paramètre $A_i$, on peut déterminer le nombre de messages à contrainte espace mémoire qui seront dans le flux $S_i$ du nœud i.
- $THT_i$ (*Token Holding Time*) est le temps de garde d'un jeton par un nœud i. Ce paramètre représente la quantité de temps pendant lequel le nœud i est autorisé à transmettre ses messages. Une sélection appropriée de ce paramètre est nécessaire pour borner le temps d'accès au canal de communication.

On suppose que tous les nœuds sont similaires et disposent des mêmes capacités de calcul et de communication. Par conséquent, les paramètres suivants sont communs à tous les nœuds :

- D est l'heure limite relative d'arrivée des messages à contraintes temps réel. Cette valeur représente la quantité de temps maximale qui peut s'écouler entre le temps de génération d'un évènement et l'heure de son arrivée à la destination. L'arrivée du $j^{ème}$ message qui contient des évènements générés à $t_{i,j}$ doit être achevée avant l'heure marquée par l'instant $t_{i,j} + D$.
- $\delta$ représente le temps nécessaire pour la transmission d'un message sur le canal partagé.

*Formulation des contraintes*

Ce paragraphe met en équation les contraintes temps-réel énoncées précédemment dans le cahier des charges ;

- Contrainte de l'heure limite : cette contrainte déclare que chaque message doit être transmis avant l'heure limite. Formellement, si $s_{i,j}$ est le temps auquel la

transmission du j$^{ème}$ (avec j = 1, 2,…) message du flux $S_i$ est achevée, la contrainte de l'heure limite impose que, pour i =1,…, n ;

$$s_{i,j} \leq t_{i,j} + D \tag{4.1}$$

ou $t_{i,j}$ est l'heure d'arrivée et D l'heure limite.

- Contrainte d'allocation de bande passante : cette contrainte veille à ce que la somme des proportions de la bande passante allouées aux nœuds du réseau ne dépasse pas la bande passante disponible du réseau. Formellement, la somme des THTs dans tous les nœuds doit être inférieure ou égale à TTRT moins le temps de transition de jeton,

$$\sum_{i=1}^{n} THT_i \leq TTRT - \tau \tag{4.2}$$

*Nombre de visites du jeton*

Pour garantir l'heure limite de l'arrivée d'un évènement à un nœud donné, il est nécessaire de connaître le nombre de visites du jeton à ce nœud pendant une période donnée. Dans la littérature, il existe plusieurs études qui déterminent d'une façon théorique les propriétés temporelles des réseaux anneau à jeton (Yao, 1995). Entre autres, les travaux de Johnson et Sevicik (Sevicik, 1987 ; Pleinevaux, 2005) énoncent, sous forme de théorème généralisé, le nombre de visite de jeton à un nœud i pendant un intervalle de temps quelconque,

*"Soit l'intervalle de temps I, le jeton visitera le nœud i au moins $v_i$ fois, avec* $v_i = \left\lfloor \frac{I}{TTRT} - 1 \right\rfloor$ *(le symbole* $\lfloor x \rfloor$ *fait référence à la partie entière par valeur inférieure)*

*Dans chacune de ces visites, le nœud i peut utiliser tout le temps $THT_i$ qui lui est alloué pour transmettre ses messages".*

*Métrique de performance*

Pour jauger la performance du système, il est nécessaire d'avoir une métrique de performance appropriée. L'*utilisation effective* est une métrique largement répandue dans les systèmes temps-réel distribués. Cette métrique représente le nombre d'évènement attendus par cycle de transmission ($\delta$). Dans le contexte des réseaux de neurones impulsionnels, l'utilisation effective d'un flux de messages $S_i$ d'un nœud i peut être représentée par l'équation suivante,

$$U_i = A_i.(1 + C_i).\delta \tag{4.3}$$

où $A_i$ est la fréquence de production des messages à contraintes temps-réel et $A_i.C_i$ la fréquence de production des messages à contrainte espace-mémoire dans un nœud i. Il en résulte que $A_i.(1+C_i)$ représente la fréquence des messages sortants du nœud i. En multipliant cette entité par le temps de transmission d'un message $\delta$, on obtient l'utilisation effective $U_i$ du nœud i, qui est la valeur moyenne du pourcentage d'utilisation des ressources de communication par le nœud i.

L'utilisation effective d'un réseau composé de n nœuds est déterminée par l'équation suivante :

$$U = \sum_{i=1}^{n} A_i.(1 + C_i).\delta \tag{4.4}$$

On appelle *utilisation atteignable* $U_i$' d'un nœud i une utilisation effective de ce nœud qui garantit que tous les messages sortants d'un flux $S_i$ atteignent leur destination avant l'heure limite D. Une conséquence immédiate de cette définition énonce que tout message issu d'une utilisation effective inférieure ou égale à l'utilisation atteignable satisfait la même contrainte temps-réel. Par exemple, si un nœud possède une utilisation atteignable U'= 0.6, alors tous les messages ayant une utilisation effective $U \leq 0.6$ atteindront leur destination avant l'heure limite D.

On définit l'*utilisation atteignable du pire cas* $U_i$* la plus grande valeur des utilisations atteignables d'un nœud i. $U_i$* correspond à l'utilisation effective du plus grand nombre de messages sortants qui peuvent être générés par un nœud. En partant de l'hypothèse qui énonce que tous les nœuds du réseau sont identiques, $U_i$* demeure la même pour tous les nœuds. Il en résulte l'équation (4.5) suivante,

$$U^* = n.U_i^* \tag{4.5}$$

Cette définition permet d'évaluer l'aspect temps réel du système : si tous les messages d'un réseau atteignent leur destination avant l'heure limite dans les conditions du pire cas, alors il en sera de même pour tous les autres cas. Formellement, pour garantir les contraintes (4.1) et (4.2) dans un système temps-réel, il faut que les conditions suivantes soient satisfaites,

- $U^* \leq 1$ : la bande passante utilisée dans le pire cas ne doit pas dépasser la bande passante disponible,
- Les messages relatifs à une utilisation atteignable du pire cas U* atteignent leur destination avant l'heure limite D.

L'importance de définir l'utilisation atteignable du pire cas est reliée aux exigences fondamentales en stabilité et prédictibilité dans les systèmes distribués temps-réel. Du fait que chaque utilisation effective U d'un flux de messages donné ne dépassera jamais U*, on peut prévoir la satisfaction de la contrainte temps-réel dans le système en l'appliquant au pire cas.

L'utilisation U* fournit aussi une stabilité pour le système. Au changement des paramètres d'un flux de messages ($A_i$ et $C_i$) la valeur de U* reste inchangée puisqu'elle a été calculée en considérant les valeurs maximales de ces paramètres. Cette immunité contre la variation de l'état d'activité du réseau permet d'évaluer la contrainte temps-réel par rapport à l'utilisation U* : si l'aspect temps réel est garanti avec une utilisation U* alors il est garanti pour toutes les autres utilisations. Ceci fournit une certaine stabilité dans le fonctionnement du système contre les changements du trafic sur le canal de communication.

*Méthode intégrative pour une allocation équitable de la bande passante*

Dans ce qui suit, on abordera la description d'une méthode intégrative pour l'allocation de la bande passante offerte par le canal de communication. Le terme ″méthode intégrative″ vient du fait que la métrique d'évaluation est intégrée dans la définition du schéma d'allocation de la bande passante. Cette méthode a pour but de répartir équitablement les accès au canal partagé tout en respectant les contraintes temporelles nécessaires pour le fonctionnement en temps réel.

Les paramètres TTRT et $THT_i$ nécessitent une sélection appropriée pour assurer une circulation fluide du jeton et une bonne disponibilité de l'utilisation du canal partagé. Si un $THT_i$ alloué à un nœud i est petit, le nœud pourrait ne pas avoir assez de temps pour accéder au canal pour transmettre ses messages. Inversement, un grand $THT_i$ pourrait causer une lente circulation de jeton autour de l'anneau, qui pourrait, à son tour, être la cause de la violation

des contraintes temporelles. Dans la même logique, si le TTRT est grand, le jeton risque d'arriver trop tard au nœud en question pour transmettre ses messages.

*Sélection de $THT_i$*

La sélection de THT est une étape importante pour garantir que les messages transmis ne rateront pas leur heure limite. Les paramètres d'un nœud ($A_i$ et $C_i$) ainsi que les paramètres du réseau (TTRT, $\delta$, D, et $\tau$) doivent être les facteurs qui interviennent dans la détermination de THT.

On définit un schéma d'allocation comme un algorithme qui, ayant comme entrées les valeurs des paramètres du nœud et du réseau, produit en sortie des valeurs de THT. Formellement, soit une fonction f qui représente un schéma d'allocation,

$$f(A_1, C_1, A_2, C_2 ...., A_n, C_n, \tau, TTRT, \delta, D) = (THT_1, THT_2, ...., THT_n) \qquad (4.6)$$

L'enjeu est de garantir l'arrivée des messages à leur destination avant l'heure limite. En utilisant la propriété du nombre de visites du jeton à un nœud donné, on peut déterminer le nombre de passage du jeton pendant une période limite D. Ce nombre vaut au moins $v_i = \left\lfloor \frac{D}{TTRT} - 1 \right\rfloor$ visites. Ceci suggère que pour qu'un message au nœud i ne rate pas son heure d'arrivée limite, la valeur du paramètre $THT_i$ doit être suffisante pour envoyer le message durant l'une des $v_i$ visites. En moyenne, le nombre de messages arrivants à un nœud dans un intervalle de temps donné doit être égal ou inférieur au nombre de messages que le nœud peut transmettre durant le même intervalle de temps. Le terme $A_i.(1+C_i).\delta$ peut représenter la charge appliquée par un nœud i. Il reflète le nombre de cycles de transmission nécessaires pour l'envoi des messages. Considérons l'intervalle de temps D, le terme $A_i.(1+C_i).\delta.D$ représente la demande de trafic pour le nœud i durant la période de temps D. Cette demande de trafic doit être véhiculée sur le canal de communication dans chaque intervalle de temps de durée D. Puisqu'un nœud peut transmettre à $v_i$ reprises pendant la période D, on peut écrire l'équation suivante pour allouer le $THT_i$,

$$THT_i = \frac{A_i.(1+C_i).\delta.D}{\left\lfloor \frac{D}{TTRT} - 1 \right\rfloor} \qquad (4.7)$$

La même équation peut être réécrite en intégrant la métrique d'évaluation $U_i$ (équation (3)) dans le numérateur,

$$THT_i = \frac{U_i.D}{\left\lfloor \frac{D}{TTRT} - 1 \right\rfloor} \qquad (4.8)$$

Cette équation est appelée *schéma d'allocation de la bande passante*, ou *schéma d'allocation* tout court. La méthode utilisée pour sélectionner $THT_i$ pour chaque nœud i est appelée *méthode intégrative* : elle intègre la métrique d'évaluation $U_i$ dans le schéma d'allocation lui-même. Cette intégration a pour résultat de faire l'autoévaluation du trafic du nœud avant l'allocation de la bande passante.

Les schémas d'allocation peuvent être classés selon deux catégories : les schémas d'allocation locaux et les schémas d'allocation globaux. La différence est dans le type des informations utilisées. Un schéma d'allocation local utilise les informations disponibles localement à chaque nœud. Un schéma d'allocation global utilise non seulement les informations locales à un nœud mais aussi des informations supplémentaires relatives aux

autres nœuds. Le schéma défini par l'équation (4.7) utilise les paramètres relatifs au flux des messages sortants ($A_i$, $C_i$ et D) et les paramètres du réseau ($\tau$, $\delta$ et TTRT). Une fois fixés, les paramètres de réseau restent inchangés. Tous les nœuds disposent donc de ces paramètres. Il en résulte que le schéma d'allocation présenté par l'équation (4.7) est un schéma local.

Un schéma d'allocation local est préférable du point de vue gestion de réseau ; si un paramètre d'un flux de message dans un nœud i change, alors seulement le $THT_i$ de ce nœud qui sera recalculé. Les THTs relatifs aux autres nœuds ne changeront pas puisqu'ils sont calculés indépendamment des paramètres du nœud i. Ceci rend un schéma d'allocation flexible et utilisable dans des environnements dynamiques.

*Sélection de TTRT*

Le paramètre TTRT a un impact direct sur la circulation du jeton et la réactivité du réseau. Il doit être également le sujet d'une sélection prudente pour satisfaire les contraintes temps-réel. La méthode de sélection de TTRT que l'on propose part de l'hypothèse que tous les nœuds sont similaires et du schéma d'allocation décrit par l'équation (4.7). On aura besoin aussi de l'équation de l'utilisation atteignable du pire cas U* pour déterminer la valeur optimale de TTRT.

En régime pire cas, les nœuds génèrent une quantité maximale de messages. Par conséquent, chaque nœud i allouera et utilisera la valeur maximale de $THT_i$ ($THT_{max}$). Puisque les nœuds possèdent les mêmes propriétés temporelles, $THT_{max}$ sera le même pour tous les nœuds. L'inéquation (4.2) deviendra,

$$THT_{\max} = \frac{TTRT - \tau}{n} \tag{4.9}$$

Par conséquent, les équations (4.8) et (4.9) donnent,

$$THT_{\max} = \frac{TTRT - \tau}{n} = \frac{U_i^*.D}{\left\lfloor \frac{D}{TTRT} - 1 \right\rfloor} \tag{4.10}$$

On peut extraire $U_i$* de l'équation précédente,

$$U_i^* = \frac{TTRT - \tau}{n.D}\left\lfloor \frac{D}{TTRT} - 1 \right\rfloor \tag{4.11}$$

En utilisant l'équation (4.5), on peut déduire U* de tout le réseau,

$$U^* = \frac{TTRT - \tau}{D}\left\lfloor \frac{D}{TTRT} - 1 \right\rfloor \tag{4.12}$$

Pour obtenir la valeur optimale de TTRT, on doit chercher la valeur maximale de U*. Maximiser la fonction U* a pour conséquence de borner le trafic temps-réel par une utilisation maximale. Il serait donc possible d'extraire la valeur du paramètre TTRT qui correspond au trafic maximal. Cette valeur permettra de transmettre tous les messages du pire cas et donc elle englobera les opérations de transmission relatives aux autres cas.

Maximiser la fonction U* revient tout d'abord à maximiser la valeur de la partie entière inférieure du terme $\frac{D}{TTRT} - 1$. Pour se faire, il faut que le terme $\frac{D}{TTRT}$ soit un entier. Supposons que le résultat de division euclidienne de D par TTRT donne un entier qu'on nommera q. On aura donc D = q.TTRT avec $q \in N$*, et U* deviendra,

$$U^* = 1 - \frac{\tau}{TTRT} - \frac{TTRT - \tau}{D} \qquad (4.13)$$

La seconde étape pour maximiser U* est de chercher le maximum absolu de l'équation (4.13). Cette équation calcule la valeur de U* en fonction de D, TTRT et τ. Le paramètre D est fixé par l'utilisateur. D doit être inclus dans une gamme de valeurs qui garantit le fonctionnement temps-réel du système. Le paramètre τ est une constante qui découle des caractéristiques temporelles de la vitesse de circulation du jeton. En conclusion, la fonction U* est fonction de la variable TTRT. En calculant la dérivée de l'équation (4.13) par rapport à la variable TTRT, on trouve un maximum unique qui vaut,

$$TTRT = \sqrt{\tau . D} \qquad (4.14)$$

Cette solution correspond au maximum de la fonction U*. La figure 4.2 montre les différentes formes de graphes obtenus de la fonction U* en fonction de TTRT. Le paramètre D (ou heure limite) change pour chaque courbe. Le temps de transition du jeton τ est fixé à 0.8 µs.

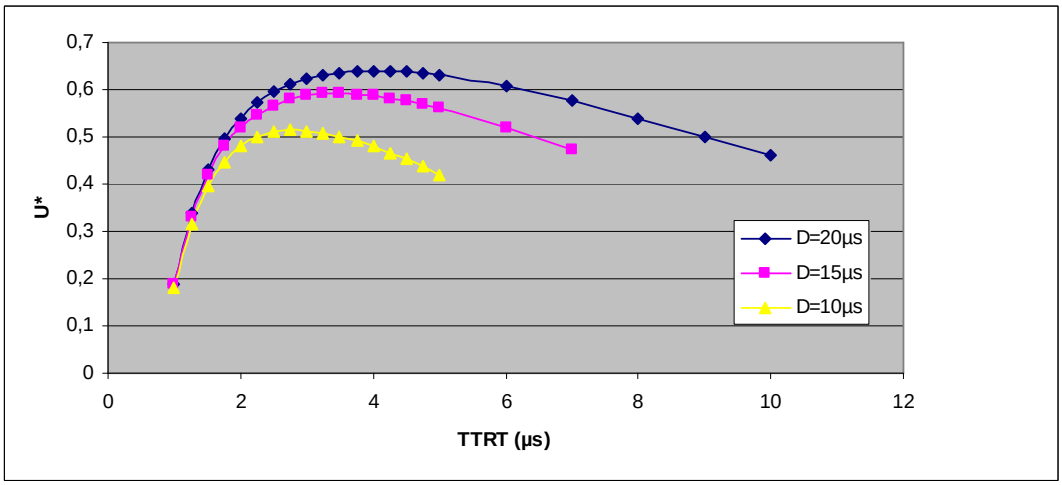



Figure 4.2. Variation de la fonction U* en fonction de TTRT et de D.

*Sélection arbitraire de D*

La sélection du paramètre D, ou heure relative limite d'arrivée des messages, est laissée à l'utilisateur. La contrainte temps-réel de l'application impose une valeur pour ce paramètre. Cependant, pour que le choix de D soit cohérent avec la sélection des autres paramètres, il faut que les conditions suivantes soient respectées,

- La valeur de D doit être au moins deux fois plus grande que la valeur de TTRT ($D \geq 2TTRT$). Cette condition garantit que chaque nœud a au moins une chance d'envoyer ses messages.
- Le résultat de la division euclidienne D/TTRT est un entier $q \in IN^* - \{1\}$.

Dans la figure 4.2, la valeur maximale de U* est atteinte à TTRT = 4 µs et D = 20 µs. Ceci est confirmé par l'équation (4.14) et vérifie les conditions de choix du paramètre D (q = 5). Cependant, si on calcule la valeur maximale de U* pour la valeur de D = 15 µs, d'après l'équation (4.14) on aura 3.46 µs qui n'est pas une valeur entière de q dans IN* – {1}. Dans ce cas, on prendra la partie entière supérieure de la valeur de TTRT, à savoir dans ce cas 4.

Dans le cas où D = 10 µs, la valeur maximale de U* est obtenue à TTRT = 2.82 µs. De même ici, on prendra TTRT = 3 µs.

### 2.3 Le protocole de communication : vue en couches

Pour mettre en œuvre la méthode développée, on a défini un protocole de communication qui se base sur un algorithme d'accès inspiré du schéma d'allocation décrit par l'équation (4.7). Pour gérer la complexité du projet, le protocole de communication a été découpé en 4 couches comme l'indique la figure 4.3. Chaque couche dispose d'un certain nombre de services prêts à être utilisés par les couches supérieures. La communication entre les couches se fait dans les deux sens. La couche supérieure commande la couche immédiatement inférieure. La couche inférieure propose des services à la couche immédiatement supérieure.

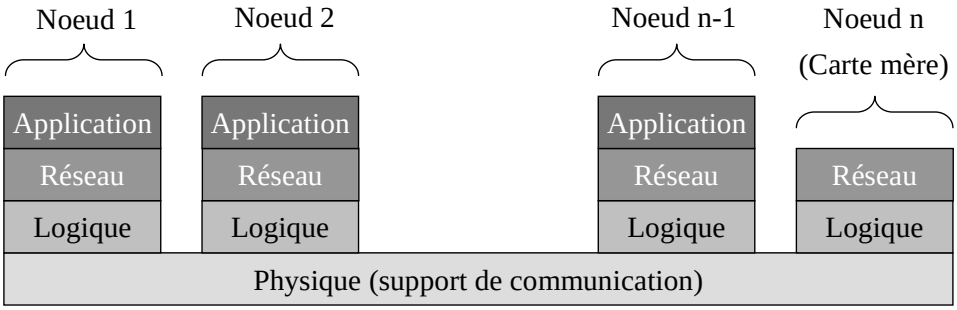



Figure 4.3. Représentation en couches du système PAX3

*Couche physique*

La couche physique, commune à tous les nœuds, offre des services de communication bas niveau. Elle est caractérisée par la bande passante, la structure du bus, le délai de propagation de l'information, etc.

Le bus de communication est conçu pour un trafic numérique et contient 64 fils conducteurs disposés en parallèle. Chaque fil supporte un débit de 25 Mbits/s. Les 64 fils sont accessibles par les E/S des FPGAs de chaque carte. La répartition des ressources de bus est donnée dans l'annexe B.

Un fil particulier est disposé en forme d'anneau autour des cartes. Il assure une communication point-à-point entre deux cartes successives branchées sur le bus de communication. Le débit binaire de transmission peut atteindre 50 Mbits/s. Il servira pour véhiculer le jeton entre les cartes.

*Couche logique*

La couche logique vient juste après la couche physique, et elle est spécifique à chaque nœud. Elle est responsable de l'accès en entrée ou en sortie des données logiques. Elle peut offrir des services d'émission/réception des données, mode de codage de l'information, encapsulation/décapsulation des trames, etc.

Le mode de transmission (asynchrone ou synchrone, série ou parallèle) est également décidé au niveau de cette couche. Le choix entre un mode ou un autre dépend des exigences de l'application cible. Dans notre cas, l'application consiste à construire des réseaux de neurones impulsionnels fonctionnant en temps réel. Cette application exige des délais de transmission faibles et une communication ″ spontanée″. Le mode de transmission asynchrone est celui qui répond le mieux à ces exigences ; à chaque fois qu'un potentiel d'action est généré il est émis directement sans attendre une référence de synchronisation.

Le mode série consiste à envoyer une donnée bit par bit en utilisant un seul fil. Pour envoyer une donnée de n bits, il faudra n cycles d'horloge. Par contre, l'envoi en parallèle consomme un seul cycle d'horloge pour envoyer une donnée de n bits. En contrepartie, il utilise une nappe de n fils. Le mode série est utilisé généralement dans les applications qui ne demandent pas un grand débit instantané, tels que les applications qui produisent des rafales de trafic sur le bus de communication. Le mode parallèle possède un débit instantané important, mais pour une courte durée. Son utilisation est souhaitée dans les applications à débit en rafales espacées dans le temps ; une caractéristique inhérente aux réseaux de neurones. Le mode asynchrone/parallèle est donc retenu.

En fonctionnant en mode asynchrone, la communication inter-cartes dans le système PAX3 n'est pas cadencée par une horloge. Chaque carte se base sur sa propre horloge pour recevoir ou envoyer les données. Malgré le fait que les horloges des cartes fonctionnent toutes à 100MHz, un déphasage entre les cycles des horloges est cependant très probable. Par conséquent, l'état des données envoyées par une carte émettrice peut changer avant que la carte réceptrice ne les lise, ou au contraire, la carte réceptrice peut lire deux fois les mêmes données. Ces problèmes de déphasages possèdent deux origines,

- Les débuts et fins des cycles des horloges des différentes cartes ne coïncident pas,
- La dispersion inhérente aux oscillateurs qui génèrent les cycles des horloges : un oscillateur peut être plus rapide ou plus lent qu'un autre oscillateur. Cette dispersion est quantifiée par les constructeurs du composant oscillateur. Pour les cartes *Ekero*, l'erreur dans un cycle d'horloge peut atteindre ±100 ppm[1].

Pour garantir que toutes les cartes lisent la bonne donnée, il faut tenir compte de l'incertitude sur la fréquence d'oscillation. La figure 4.4 illustre ce fait. Les traits interrompus courts représentent la cadence de l'horloge de la carte réceptrice, alors que les traits continus représentent la cadence de l'horloge de la carte émettrice. Les deux horloges oscillent à la même fréquence. Dans cette figure, on suppose que l'horloge réceptrice est plus lente que l'horloge émettrice à cause de l'incertitude sur les fréquences des deux horloges. Cette incertitude induit un déphasage Δt qui s'accumule en cours du temps.

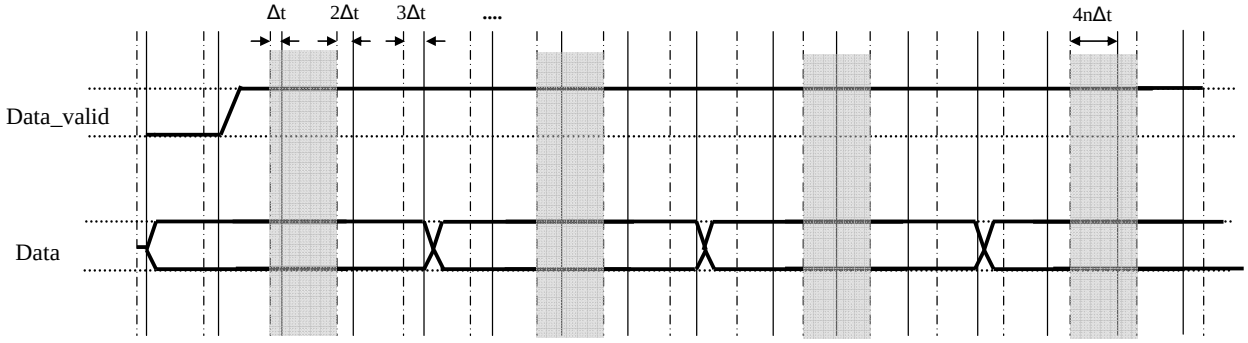



Figure 4.4. Illustration de l'incertitude sur la fréquence d'oscillation dans la communication parallèle asynchrone.

Le principe de codage des données est le suivant : le signal *Data_valid* informe le récepteur de la présence des données sur le bus. Le signal *Data* contient la valeur des données codée sur 64 bits. Pour être sûr que les bonnes données seront lues par la carte réceptrice,

[1] Part Per Million.

chaque donnée reste sans changement pendant trois cycles de l'horloge émettrice. Elle sera par suite capturée au niveau du récepteur pendant le cycle coloré en gris. L'erreur de déphasage ne doit pas dépasser un cycle d'horloge pour être sûr de l'intégrité des données.

L'erreur s'accumule après chaque cycle d'horloge pour atteindre les $4n\Delta t$ au niveau du $n^{ième}$ cycle de transmission. Pour calculer combien de cycles de transmission successifs, on calcule $4n_{max}\Delta t < T$, avec T la période des deux horloges. Cette période vaut 10 ns (±100 ppm) dans le cas des cartes du système PAX3. Ce qui donne $n_{max} < T/(4\Delta t)$, où $T/\Delta t = 10^6/200$ (100 + 100 ppm). Il est donc possible d'envoyer des données sur 1250 cycles de transmission successifs sans risque de décalage des données.

*Couche réseau*

La couche réseau est la couche où le schéma d'allocation de la bande passante est implémenté. Cette couche commandera les couches supérieures (logique et physique) pour donner l'ordre d'envoyer ou recevoir des données. Le schéma d'allocation décrit par l'équation (7) est implémenté à ce niveau. A côté du schéma d'allocation, il faut définir un algorithme d'accès qui organise l'accès au canal de communication partagé.

L'algorithme d'accès est une combinaison de décompteurs qui représentent les quantités de temps des paramètres de réseau TTRT et $THT_i$. Dans chaque nœud, il existe deux types de compteurs,

- $TRT_i$ (*Token Rotation timer*) : ce compteur est initialisé à la valeur sélectionnée du paramètre TTRT. Il est décrémenté en permanence jusqu'à ce qu'il expire ($TRT_i = 0$). Ce compteur sera remis à sa valeur initiale (TTRT) à chaque passage du jeton dans le nœud i.
- $H_i$ (*Token Holding Timer*) : Ce compteur est utilisé pour contrôler le temps de transmission de chaque nœud. Il est initialisé à $THT_i$ à chaque fois que le jeton arrive au nœud i.

Les paramètres de réseau sont supposés déjà fixés. Le seul paramètre qui change durant une simulation est $THT_i$. Ce paramètre dépend du nombre de messages sortants dans chaque nœud. Il est donc calculé en permanence en fonction des activités des nœuds.

Le principe de l'algorithme est simple puisque tous les paramètres sont déjà définis, et donc les compteurs sont bien initialisés. A chaque fois que le nœud i reçoit le jeton, le compteur $TRT_i$ est initialisé à TTRT. Le nœud a la permission d'envoyer des messages durant le temps de possession du jeton ($THT_i$) calculé selon le schéma d'allocation de l'équation (4.7). La priorité est donnée aux messages à contrainte temps-réel pour être envoyés en premier. S'il n'y a pas de message à contrainte temps-réel ou s'ils sont tous transmis, le nœud commence à envoyer les messages à contrainte espace-mémoire jusqu'à ce qu'il épuise tous les messages ou que le $THT_i$ expire ($H_i = 0$). Un nœud libère un jeton s'il n'y a plus de messages à transmettre ou si le temps d'allocation maximal ($THT_{max}$) est atteint. Dans le cas où le compteur $TRT_i$ expire ($TRT_i = 0$) et où le nœud i n'a pas encore reçu le jeton, les contraintes temps-réel du système deviennent menacées et l'heure limite de l'arrivée des messages risque d'être dépassée. La simulation est donc immédiatement interrompue et un message d'erreur est expédié pour informer l'utilisateur de la production d'une erreur de fonctionnement. L'erreur peut provenir de plusieurs raisons, entre autres, un nœud qui tombe en panne ou un problème de fonctionnement sur l'anneau à jeton. La figure 4.5 illustre l'opération de l'algorithme d'accès.

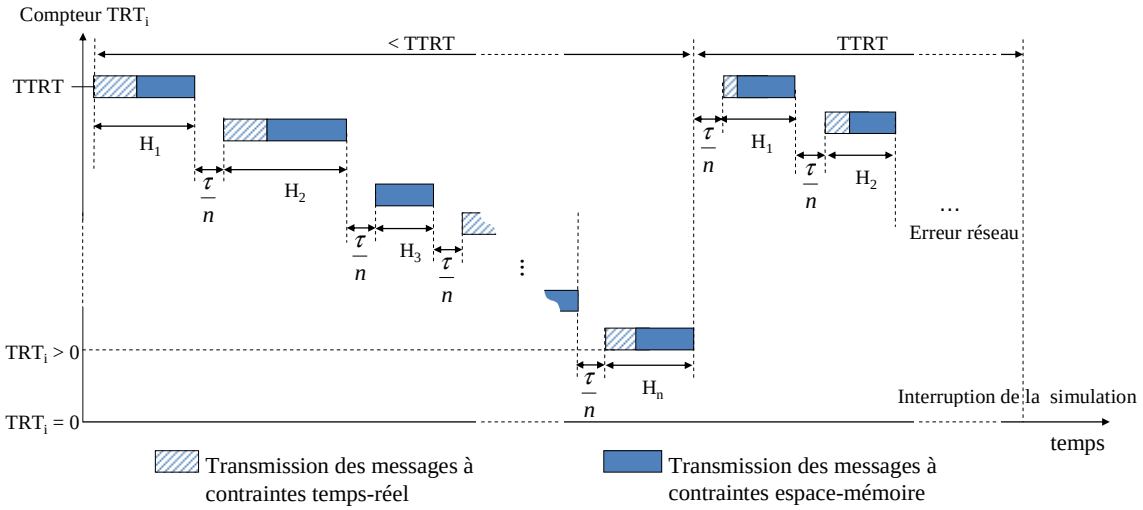



Figure 4.5. Illustration de la circulation du jeton et transmission des messages d'un nœud i.

*Couche application*

La couche application n'est pas concernée par la gestion de l'accès au support de communication (qui lui est transparente). Par contre, elle utilise ce qui a été réalisé par les couches inférieures pour communiquer ses messages. Les messages sont générés par des applications neuronales implémentées au niveau de cette couche, tels que le calcul de la plasticité et l'émulation d'un type de neurone.

Les couches d'application de chaque nœud communiquent par l'envoi et la réception des messages. Ce qui leur permet de partager les informations et donc de distribuer le calcul entre nœuds. La section 3 portera sur l'application de la plasticité distribuée dans le système PAX3, ce qui servira d'un bon exemple du rôle de la couche application.

### 2.4 La validation expérimentale

La méthode intégrative d'allocation de la bande passante définie dans la section précédente utilise le pire cas pour garantir les contraintes temporelles du système. Expérimentalement, le pire cas correspond au plus grand nombre de messages que le système PAX3 peut générer. Le nombre de messages dépend directement de l'activité des neurones et de la connectivité du réseau.

*Définition du pire cas pour le système PAX3*

Dans les réseaux de neurones impulsionnels, il y a, *a priori*, une grande probabilité que les neurones qui interagirent ensemble ne produisent pas des potentiels d'action au même instant. Une activité simultanée peut cependant se produire. Combinée avec une connectivité complète (*all-to-all*), un taux d'activité neuronale simultanée conduit au pire cas. Cette situation inonde le système avec un grand nombre d'évènements en un intervalle de temps court. Par conséquent, la demande en ressources de communication augmente très rapidement et crée des rafales de données à transmettre. L'algorithme d'accès doit faire face à cette situation en absorbant ces rafales dans le trafic de communication (Belhadj, 2010).

*Implémentation du banc de tests pour le pire cas*

Dans le but de valider le fonctionnement du protocole de communication, on a implémenté un banc de test qui sert à stresser le système en le mettant dans les conditions du pire cas. Le système contient 6 cartes filles et une carte mère. Il émule 120 neurones répartis sur les cartes fille. La figure 4.6 illustre cette implémentation. Les blocs *ASICgen* remplacent

les neurones analogiques des cartes fille et génèrent des messages à contraintes temps-réel (ou *MRT*) générés avec une fréquence de 100 Hz. Les blocs *STDPgen* simulent le taux de génération des messages à contraintes espace-mémoire (ou *MEM*) pour une connectivité complète. Le système est périodiquement soumis au pire cas avec une période de 10 ms. Ceci est équivalent à générer 20 MRT par carte toutes les 10 ms et 3400 (120*20+20*50) MEM par carte toutes les10 ms.

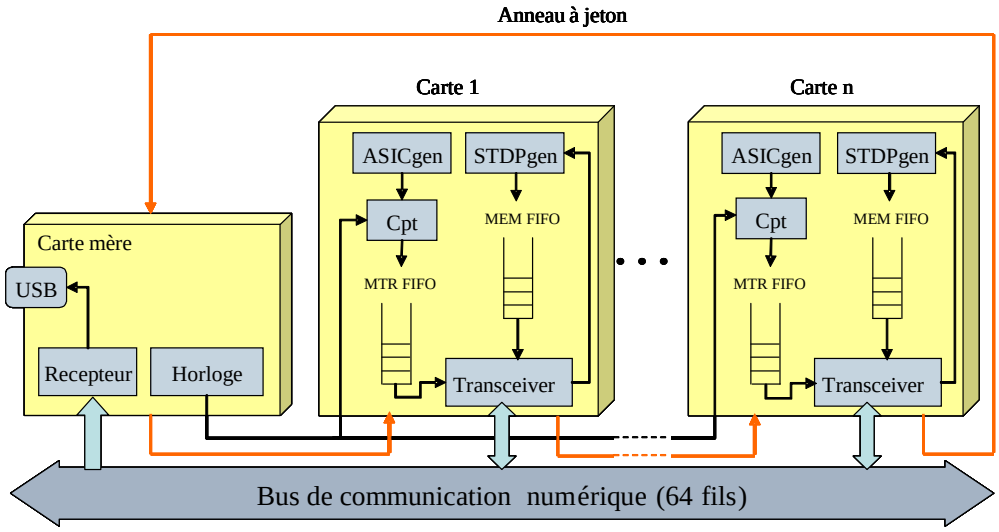



Figure 4.6. Architecture du banc de test pour la validation du protocole de communication

Une horloge commune cadencée par la carte mère sert à unifier les compteurs des instants de génération des MRT de toutes les cartes fille. Le module *émetteur-récepteur* (*transceiver*) implémente la couche logique du protocole de communication et agit comme une interface bidirectionnelle entre le bus de communication et les cartes. Les messages transmis sont reçus par la carte mère qui classifie les MRT selon leur heure d'arrivée pour vérifier la validité de la contrainte temps-réel et compte le nombre des MEM pour vérifier l'intégrité des messages transmis.

Le temps de transmission d'un message $\delta$ vaut 40 ns et le temps de transition du jeton $\tau$ vaut 0.56 µs. Les autres paramètres dérivent du choix du paramètre D ; quand l'utilisateur fixe une date limite D, TTRT et $THT_i$ sont déterminés respectivement par les équations (4.14) et (4.7).

Une série de simulations qui durent une minute ont été réalisées dans les conditions du pire cas en modifiant à chaque fois l'heure limite D. La figure 4.7 expose le nombre des MRT qui ont atteint leur destination en fonction du temps qu'ils ont pris pour arriver à la carte mère. Un tel retard est exprimé en microsecondes et est déterminé en calculant la différence entre l'heure de génération des potentiels d'action et l'heure d'arrivée de ces derniers à la carte mère. Plusieurs valeurs de l'heure limite D sont considérées ; quand la date limite devient de plus en plus petite, la moyenne des retards augmente. Ceci peut être déduit à partir des équations (4.7) et (4.14) où les valeurs de $THT_i$ et TTRT diminuent : le trafic temps-réel est étalé sur une plus longue période de temps.

Pour les simulations (a) et (b) tous les MRT satisfont la contrainte temps-réel en marquant respectivement des retards inférieurs à 10 et 5 µs. Cependant, dans les simulations (c) et (d) les MRT dépassent les heures limites fixés par l'utilisateur (4 µs et 3 µs respectivement). Ceci

est traduit par la présence des messages qui sont arrivés à la 5ème et 6ème microsecondes. Par conséquent, la simulation (b) présente la valeur expérimentale de l'heure limite minimale que l'utilisateur peut arbitrairement fixer, soit $D_{min\_exp} = 5$ µs.

Théoriquement, la valeur de $D_{min}$ peut être déterminée en remarquant que $D \geq (k+1).TTRT$, où $k$ est un entier qui représente le nombre requis de visites d'un jeton à un nœud pour transmettre tous les messages MRT dans les conditions du pire cas. Formellement, l'entier $k$ peut être déterminé par le rapport de la demande en bande passante pendant la durée TTRT (U*.TTRT) avec $THT_{max}$.

En résumé, le système garantit l'arrivée de tous les messages avant l'heure limite quelque soit $D \geq D_{min}$.

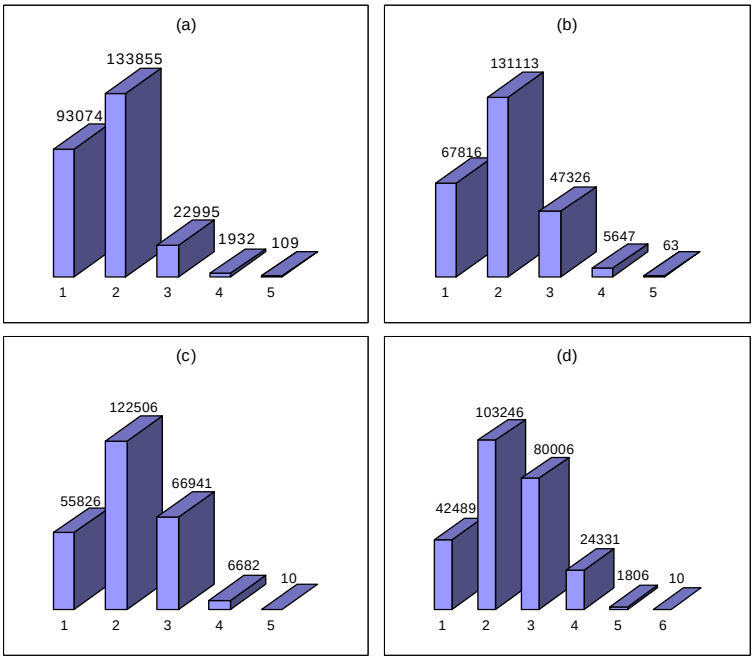



Figure 4.7. Retard dans le temps d'arrivée des messages temps-réel (ou MRT) à leur destination. (a) D = 10 µs, (b) D = 5 µs, (c) D = 4 µs, (d) D = 3 µs.

*Combien de cartes peut-on insérer dans le système ?*

La version actuelle du système contient 6 cartes filles et une carte mère. Le nombre de cartes fille est susceptible d'augmenter pour permettre la simulation des réseaux de neurones plus grands. La question qui se pose dans ce cas là est : combien de cartes peut-on brancher sur le système sans altérer le fonctionnement temps-réel du protocole de communication ? Autrement dit, quel est l'impact du nombre de cartes n sur l'heure limite D et le temps d'allocation de la bande passante THT ?

Le nombre de cartes n influe d'une façon directe sur le paramètre $\tau$ (*Token Walk Time*). Ce paramètre représente la somme des temps que met le jeton pour transiter d'une carte à une autre pendant un tour complet. Supposons que T soit la durée de transmission du jeton d'une carte à la suivante, on a alors $\tau = n.T$. D'autre part, l'envoi des messages dépend de la valeur du paramètre $THT_{max}$ qui reflète le nombre maximal des messages sortants. Pour un bon fonctionnement du protocole de communication, il faut que la valeur de $THT_{max}$ soit suffisante pour envoyer tous les messages avant l'heure limite D. Les équations (4.9) et (4.14) donnent,

$$THT_{\max} = \frac{TTRT - \tau}{n} = \frac{\sqrt{n.T.D} - n.T}{n} \qquad (4.15)$$

Cette formule exprime le paramètre $THT_{max}$ en fonction du nombre des cartes n. Elle reflète le résultat de l'augmentation du nombre des cartes sur le temps d'allocation de la bande passante d'un nœud. La figure 4.8 trace les courbes relatives à cette équation en fixant T = 0.08 µs pour plusieurs valeurs du paramètre D. Le nombre de cartes maximal que l'on considère ici est 50. D'une façon générale, la valeur $THT_{max}$ diminue quand le nombre de cartes augmente. Cette diminution est fonction du paramètre D ; elle est moins importante lorsque D est grand.

Intuitivement, le nombre de cartes dans le système engendre plus d'obstacles devant la rotation du jeton, ce qui ajoute un temps de rotation supplémentaire. Ce temps est soustrait du temps alloué pour transmettre les messages des nœuds. Par conséquent, plus on a de cartes dans le système plus le temps de rotation du jeton devient plus grand et plus le temps de transmission des messages diminue. La garantie de l'aspect temps-réel est fonction de l'heure limite préfixée par l'utilisateur. C'est à l'utilisateur donc de quantifier ses besoins en fonction du nombre de cartes, de l'heure limite et du temps d'allocation de la bande passante.

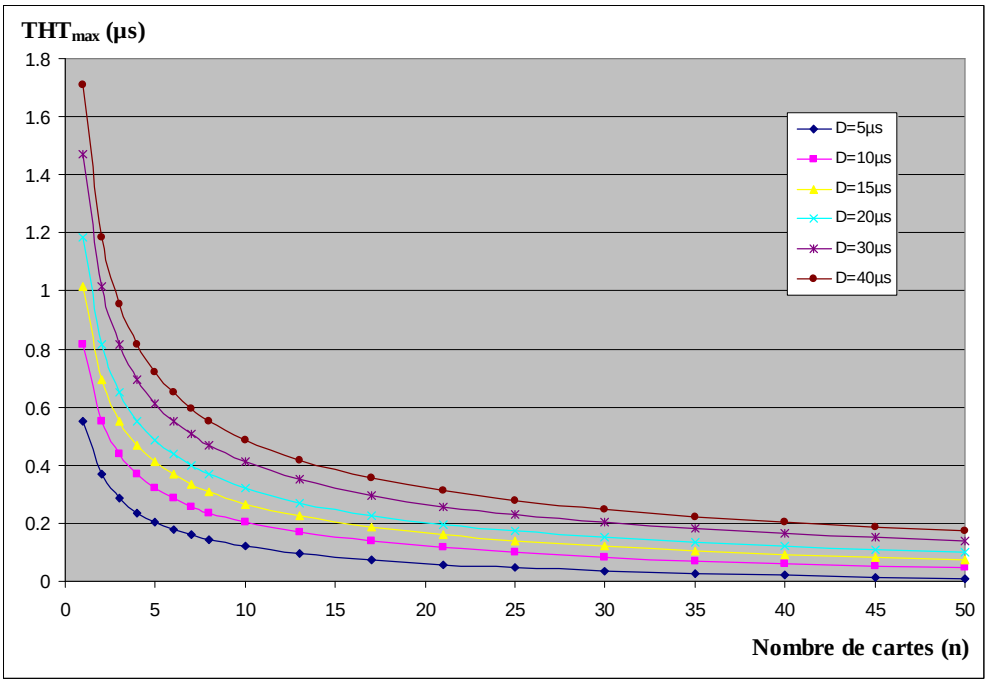



Figure 4.8. Evolution de la valeur de $THT_{max}$ en fonction du nombre des cartes n et du paramètre D.

## 3 La plasticité distribuée

Rappelons au début de cette section que l'ensemble du système loge dans un rack qui permet de relier 21 cartes sur un fond de panier commun. Chaque carte représente une sous-unité de calcul qui gère en local 20 neurones, sauf la carte mère (ou maître) qui possède des fonctions de contrôle et de relais. Chaque sous-unité a un accès à l'activité de l'ensemble de réseau. La carte mère contrôle cet accès qui s'opère selon le protocole de communication temps-réel. La section 4 du chapitre 2 détaille l'architecture du système conçue pour gérer la plasticité d'un réseau de 500 neurones.

Le calcul de la plasticité du réseau est réparti sur plusieurs sous-unités de calcul. La séparation physique des ressources de calcul doit être accompagnée par une cohérence dans le traitement global de la connectivité et la plasticité.

### 3.1 La distribution du calcul

Le but de la distribution du calcul de la plasticité est d'utiliser les connexions plastiques dispersées dans plusieurs sous-unités de calcul et de les faire fonctionner comme si elles appartenaient à un même grand réseau. Les matrices de poids, de plasticité et de la polarité synaptique représentent les données relatives aux connexions synaptiques. Le stockage de ces données dans les mémoires des sous-unités est le point clé pour une répartition équilibrée du traitement. Le calcul de la plasticité d'une connexion s'effectuera dans la sous-unité où sont stockées ses données. Dans le cas où le calcul est fait ailleurs, il serait nécessaire de transférer des données sur le bus de communication pour effectuer un calcul. Puis, renvoyer de nouveau les résultats du calcul à l'unité d'origine. Ceci alourdirait alors la quantité d'informations à envoyer sur le bus de communication et conduirait à une congestion rapide de ce dernier. Il est donc nécessaire que l'emplacement du calcul suive l'emplacement des données.

L'action des connexions synaptiques est représentée par leur poids. Ces poids modulent les stimulations postsynaptiques afin d'exciter ou d'inhiber les neurones cibles. L'emplacement des poids est, par conséquent, lié à l'emplacement des neurones postsynaptiques. Cela signifie que chaque sous-unité mémorise les poids synaptiques de l'ensemble des connexions *externes afférentes*, en plus des connexions *locales*. Ce qui impose un calcul in situ des poids correspondant à ces connexions. La figure 4.9 illustre la représentation matricielle des poids synaptiques distribués sur n sous-unités de calcul du système PAX3. Il en est de même pour la distribution des données dans les matrices de plasticité et de polarité synaptique.

Dans le système PAX3, le réseau peut atteindre les 500 neurones répartis sur des cartes contenant chacune 20 neurones. Chaque sous-unité est en charge de calculer 480*20 connexions afférentes externes et $20^2$ connexions locales. Soit un total de 10000 connexions synaptiques possibles par unité de calcul.

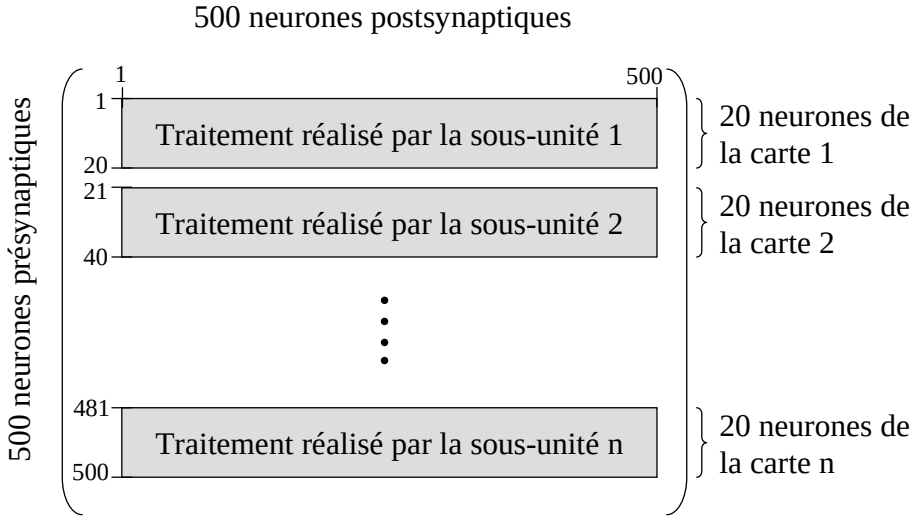



Figure 4.9. Répartition du traitement des connexions plastiques sur les sous-unités du système PAX3.

Encore une fois, on utilisera le pire cas pour spécifier les exigences de calcul dans chaque sous-unité. Le pire cas arrive quand le réseau est complètement connecté (all-to-all) et les neurones génèrent des potentiels d'action avec leur fréquence maximale (100 Hz). Les sous-unités seront amenées dans ce cas à calculer 10000 connexions plastiques chaque 10 ms. Si une sous-unité satisfait les contraintes temps-réel relatives au calcul de la plasticité et dispose des ressources matérielles nécessaires pour le fonctionnement durant le pire cas, alors elle est capable de satisfaire toutes les contraintes dans tous les autres scénarii possibles.

### 3.2 L'algorithme de plasticité distribué

On utilisera le modèle de plasticité décrit par l'équation (1.8) de la section 3 du chapitre 1 pour implémenter la plasticité dans le système PAX3. Déterminée à partir des constatations expérimentales, cette équation modélise les règles d'évolution d'un poids synaptique d'une paire de neurones isolés. Elle fait intervenir les potentiels d'action pré et postsynaptiques pour déterminer la variation du poids.

La gestion centralisée de la plasticité de tout le réseau ne pose pas de problèmes de compatibilité avec l'hypothèse implicite de la localisation des neurones dans la formule (1.8). Ceci est le cas dans le système PAX2 où tout le réseau est géré par une seule unité de calcul (voir Chapitre 3). Par contre, l'architecture PAX3 sépare la gestion des neurones sur plusieurs sous-unités de calcul. D'où le besoin d'adapter l'équation (1.8) au contexte du calcul distribué.

Comme il a été mentionné dans la section précédente, chaque sous-unité est en charge de la gestion de toutes les connexions locales et les connexions externes afférentes. Cela signifie que les évènements venant de l'extérieur (des autres sous-unités) sont forcément des potentiels d'action présynaptiques, et les évènements venant de l'intérieur (de la même carte) peuvent être des potentiels d'action pré ou postsynaptiques. Notons l'ensemble des neurones externes par $N_{ext}$, et l'ensemble des neurones internes par $N_{int}$. La formule (1.8) devient dans ce cas un système à deux équations,

$$\begin{cases} \dfrac{dw_{ij}}{dt} = \varepsilon_i\ \varepsilon_j \left\{ (w_{LTP} - w_{ij}) \sum_{t_i} P\left[(t - t_j^{last}(t)\right] \delta(t - t_i) \right. \\ \qquad \left. - (w_{ij} - w_{LTD}) \sum_{t_j} Q\left[(t - t_i^{last}(t)\right] \delta(t - t_j) \right\} & \forall i, j \in N_{\text{int}} \\ \dfrac{dw_{ij}}{dt} = -\varepsilon_i\ \varepsilon_j (w_{ij} - w_{LTD}) \sum_{t_j} Q\left[(t - t_i^{last}(t)\right] \delta(t - t_j) & \forall j \in N_{ext} \quad et \quad \forall i \in N_{\text{int}} \end{cases} \tag{4.16}$$

La fonction *P(t)* qui calcule la potentiation est annulée par la fonction de Dirac tant qu'il n'y a pas un potentiel d'action postsynaptique qui arrive. Or, tous les évènements externes sont des potentiels d'action présynaptiques. Par conséquent, le terme de la potentiation restera annulé en permanence dans les cas où le neurone présynaptique *j* fait partie de l'ensemble $N_{ext}$.

L'algorithme de plasticité relatif au système PAX3 se base sur les équations du système (4.16) pour définir les étapes de calcul. Les étapes de l'algorithme seront activées selon le type de potentiel d'action qui arrive à l'unité de calcul. La figure 4.10 résume les principales étapes de déroulement du calcul de la plasticité dans chacune des sous-unités du système PAX3.

```
Etape 0 : soient Npre = {Next∪Nint} et Npost = {Nint}
          Aller à l'étape 1.

Etape 1 : si un potentiel d'action e arrive alors
             Identifier e
             si e ∈ Next alors aller à l'étape 2.
             sinon (e ∈ Nint) aller à l'étape 3.
             fsi
          fsi

Etape 2 : si Npost ≠ {∅} alors
             Choisir un élément npost de Npost
             Calculer le poids w'(e,npost)=(w(e,npost)-wLTD).Q.εe.εnpost
             Npost = Npost−{npost}
          sinon aller à l'étape 0.
          fsi

Etape 3 : si Npre ≠ {∅} alors
             Choisir un élément npre de Npre
             Calculer le poids w'(npre,e)=(wLTP-w(npre,e)).Q.εe.εnpre
             Npost = Npre−{npre}
          sinon si Npost ≠ {∅} alors
             Choisir un élément npost de Npost
             Calculer le poids w'(e,npost)=(w(e,npost)-wLTD).Q.εe.εnpost
             Npost = Npost−{npost}
          sinon aller à l'étape 0.
          Fsi
```

Figure 4.10. Algorithme de plasticité distribué : illustration des différentes étapes suivies par chaque sous-unité de calcul du système PAX3

### 3.3 Les méthodes et les techniques d'implémentation

Cette section détaille le fonctionnement global et les techniques utilisées pour mettre en œuvre le calcul de plasticité dans le simulateur PAX3. Les méthodes développées permettent d'établir un enchaînement cohérent du traitement qui s'effectue à l'intérieur des boucles de simulation, alors que les techniques concrétisent la réalisation effective des calculs de la plasticité et des autres fonctions neuronales.

Les explications sont fournies selon une approche *top-down*. Le protocole de communication est supposé fonctionnel sans erreurs, et est représenté par le niveau d'abstraction transactionnel[1].

*Fonctionnement général*

Le fonctionnement général du système est déterminé par deux phases distinctes : une *phase de configuration* suivie d'une *phase de simulation*. La phase de configuration consiste à envoyer tous les paramètres dont le système a besoin pour simuler un réseau de neurones impulsionnels. La phase de simulation, quant à elle, combine le lancement des boucles de simulation et la récupération des résultats.

Dans un premier temps, l'utilisateur est amené à décrire le réseau à simuler : le nombre et les modèles des neurones qui constituent le réseau, ainsi que les connexions qui les relient. Il existe deux modèles de neurones que l'on peut utiliser dans le système PAX3 : les neurones

[1] Le niveau d'abstraction transactionnel consiste à considérer les opérations de communication comme des transactions indépendantes du temps. Les détails relatifs au fonctionnement du protocole de communication ne sont données.

inhibiteurs FS (ou *Fast Spiking*) et les neurones excitateurs RS (ou *Regular Spiking*) (Renaud, 2010). D'autres modèles sont susceptibles d'être ajoutés prochainement à la liste et enrichir la diversité des simulations dans le système PAX3. Les doctorants L. Buhry et F. Grassia travaillent dans l'objectif de mettre au point une méthode de réglage automatique des neurones qui permet de cibler un large éventail de modèles de neurones (Buhry, 2008).

Une fois que les neurones du réseau sont configurés, l'étape suivante sera de configurer la topologie du réseau. Cette étape revient à remplir les matrices des poids initiaux, de la plasticité et de la polarité synaptique. Dans la figure 4.11, on présente un exemple de spécification d'un réseau de 5 neurones. Les connexions inter-neurones sont étiquetées par les poids synaptiques initiaux. Les synapses excitatrices sont représentées par un cercle noir et les inhibitrices par un cercle blanc. Les synapses sont positionnées à proximité des neurones postsynaptiques, pour définir le sens de la connexion.

Il existe des langages de spécification des réseaux de neurones qui facilitent la définition des neurones ainsi que les connexions synaptiques pour différentes plateformes de simulation (logicielles ou matérielles). Cependant, il est tout de même nécessaire d'annoter ces langages avec les spécificités des architectures cibles.

A l'instar des autres groupes partenaires dans le projet FACETS, la configuration des systèmes PAX2 et PAX3 sera faite par un langage baptisé PyNN (Python for Neural Networks)(PyNN, 2005). L'unification du langage de spécification permet, entre autres, de simuler une même représentation d'un réseau de neurones sur plusieurs plateformes de simulation, et éventuellement de les comparer. Le Dr. A. Daouzli a développé, dans le cadre de sa thèse (2009), les primitives PyNN nécessaires pour la spécification des neurones, des synapses et de la topologie du réseau relatives à nos plateformes.

Les adresses des neurones dans le système PAX3 peuvent prendre deux formes différentes : locales ou globales. On appelle une adresse locale d'un neurone son numéro sur la carte où il est situé. Elle est représentée par un entier naturel compris entre 0 et 19. L'adresse globale référence le neurone à l'échelle du réseau. Elle attribue un numéro unique à chaque neurone indépendamment de son emplacement physique. Dans un réseau à $N$ neurones, les valeurs des adresses globales varient entre 0 et $N-1$.

L'architecture physique du système est représentée par une base de données qui fait correspondre les adresses globales aux adresses locales. Autrement dit, elle permet de définir la cartographie du réseau en partant des informations sur la répartition physique des neurones : les cartes et les ASICs qui les hébergent. Ces données sont d'une grande importance puisqu'elles permettent d'adresser les paramètres de configuration vers les bons composants du système. Elle permet également d'écarter les cartes défaillantes (ou non existantes) ainsi que les neurones non fonctionnels en mettant à jour la base de données. Cette étape est transparente à l'utilisateur final, et est réalisée manuellement jusqu'à présent.

La combinaison des informations apportées par la spécification du réseau avec les informations fournies par la base des données des localisations physiques dresse la cartographie finale du réseau. Un fichier global qui regroupe tous les paramètres nécessaires à la configuration du réseau est donc généré. L'envoi de ces paramètres vers les différents composants du système marque la fin de la phase de configuration.

L'envoi des paramètres de configuration se fait par la liaison USB qui connecte la machine hôte avec le système. Les adresses des neurones sont stockées dans des tables de conversion spécifiques à chaque carte fille, où l'adresse locale du neurone est associée à son adresse globale. L'adresse locale est utilisée dans les calculs internes aux cartes fille, alors que l'adresse globale est utilisée dans la communication inter-cartes.

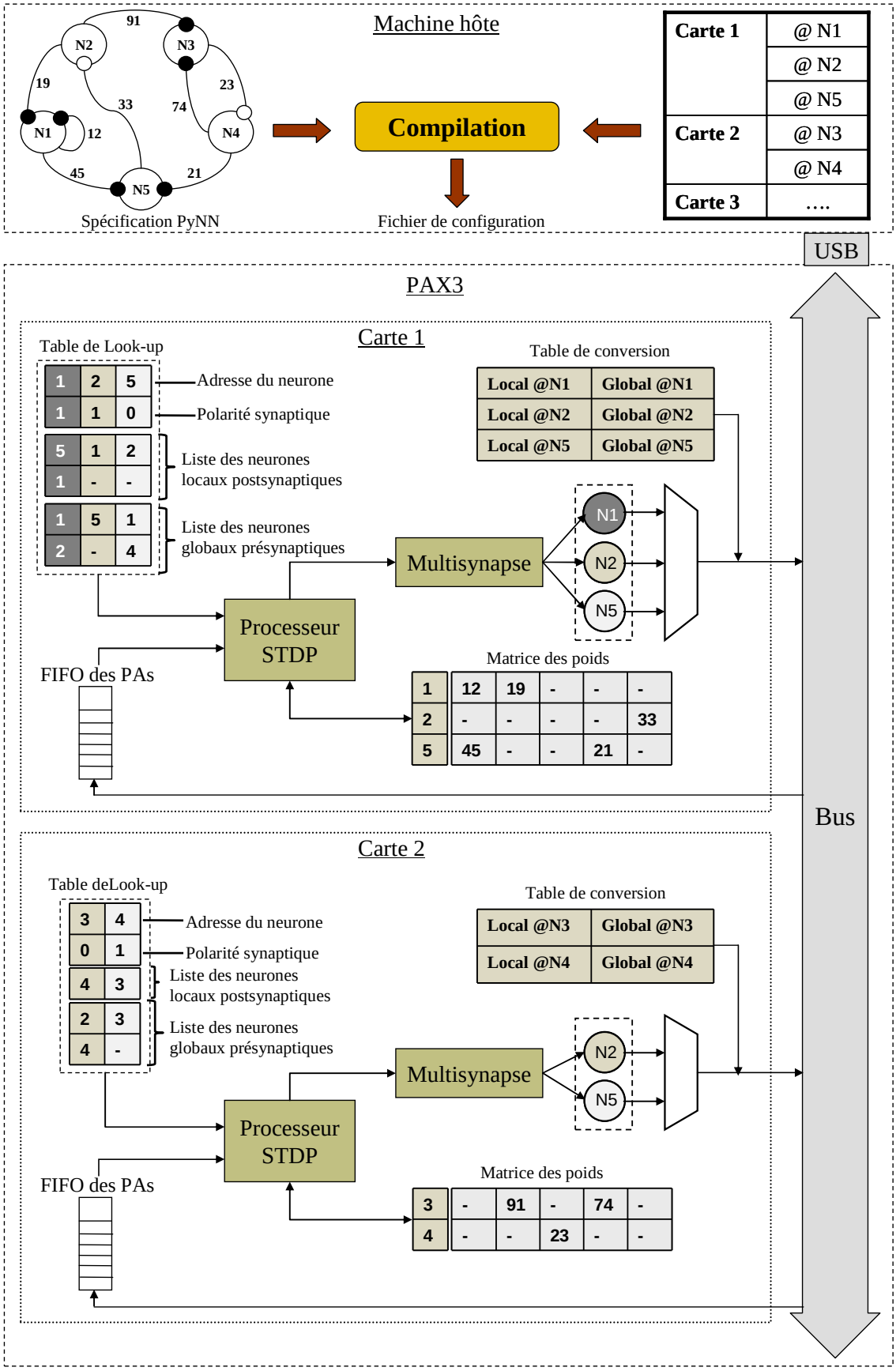



Figure 4.11. Fonctionnement général du système PAX3 : illustration sur un réseau de 5 neurones répartis sur 2 cartes.

Le système est prêt maintenant pour lancer une simulation. Un ordre spécifique envoyé par la machine hôte donne l'ordre à la carte mère de démarrer la boucle de simulation. Les neurones analogiques génèrent des potentiels d'action. Un potentiel d'action est représenté par l'adresse globale du neurone générateur à laquelle on ajoute une étiquette indiquant l'instant de génération (*timestamp*), et ce potentiel d'action est considéré comme un évènement. Chaque évènement sera diffusé sur le bus de communication de telle sorte que toutes les cartes, y compris la carte émettrice, puisse le lire. Les cartes réceptrices enregistrent tous les évènements incidents dans une FIFO.

Le *processeur de la STDP* tire les évènements de la FIFO un par un et décode chacun d'eux. L'adresse servira pour scruter les entrées des tables de *Look-up* et extraire les connexions qui relient le neurone émetteur (présynaptique) aux neurones locaux (postsynaptiques). Les tables de *Look-up* sont des mémoires RAM qui contiennent des informations concernant les connexions de chaque neurone : la liste des neurones pré et postsynaptiques, les poids, la plasticité et la polarité synaptique.

A ce stade, deux cas se présentent : si l'adresse décodée correspond à un neurone ayant une ou plusieurs connexions avec les neurones locaux, alors leur plasticité sera calculée. Sinon, l'évènement est rejeté et le processeur passe au traitement de l'évènement suivant dans la FIFO.

Lorsque le processeur retient une connexion pour un calcul de plasticité, il commence par envoyer une requête à la matrice des poids pour chercher le poids synaptique correspondant. La matrice des poids est une mémoire RAM qui contient des entrées pour chaque paire de neurones susceptible de former une connexion. Elle est initialement configurée par les poids initiaux qui seront remplacés au cours de la simulation par les poids récemment calculés.

Avant de lancer le calcul proprement dit de la plasticité d'une connexion, le processeur envoie le poids synaptique correspondant et l'adresse locale du neurone postsynaptique vers le module de la multisynapse. Ce dernier crée une impulsion dont la largeur est proportionnelle à la valeur du poids et l'envoie vers les entrées synaptiques du neurone analogique.

Juste après, le calcul de plasticité de la connexion commence. Le processeur procède selon l'algorithme de plasticité distribué décrit dans la figure 4.10. Une fois le calcul achevé, le nouveau poids est stocké à la place de l'ancien dans la matrice des poids. Ensuite, le processeur vérifie s'il existe encore des connexions relatives à l'évènement incident pour continuer le traitement, sinon, il extrait un nouvel évènement de la FIFO. Le même cycle se répète jusqu'à ce que la simulation prenne fin. La figure 4.11 illustre les éléments intervenant dans l'enchaînement de la simulation pour un exemple de réseau de 5 neurones.

*Un processeur pour la STDP*

Le schéma bloc simplifié du processeur STDP est donné dans la figure 4.12. La mémoire programme contient 10240 mots de 2 bits qui représentent les connexions du réseau. Le premier bit détermine si la connexion est plastique ou non. Le second indique la polarité de la synapse (excitatrice ou inhibitrice). La mémoire est implicitement séparée en 20 blocs de 512 entrées chacun. Un bloc contient toutes les connexions présynaptiques d'un neurone local. La sélection des connexions se fait par des adresses codées sur 14 bits : 5 bits pour déterminer le bloc et 9 bits pour extraire la connexion. Les connexions combinées avec les adresses des neurones à l'origine des potentiels d'action représentent les instructions du programme du processeur. Elles sont programmées une seule fois avant le début du calcul de la plasticité.

La mémoire de données est organisée de la même façon que la mémoire du programme. Elle stocke les poids synaptiques de toutes les connexions possibles du réseau. Un poids

synaptique est codé sur 16 bits et peut subir des modifications pendant le calcul de la plasticité.

Les évènements, ou potentiels d'action, arrivent par l'entrée PA qui permet de véhiculer les adresses des neurones qui sont à l'origine des potentiels d'action au bloc *compteur connexions*. Ce bloc permet de coder l'adresse des connexions mises en jeu par l'arrivée d'un PA. Selon la nature du neurone à l'origine de l'évènement (local ou externe), le compteur est incrémenté par 1 (scrutation des connexions présynaptiques) ou de la taille d'un bloc de mémoire (scrutation des connexions postsynaptiques). La fin du comptage est déterminée par les valeurs de deux *registres de configuration* qui indiquent respectivement le nombre des neurones mis en jeu dans le système global et le nombre des neurones locaux.

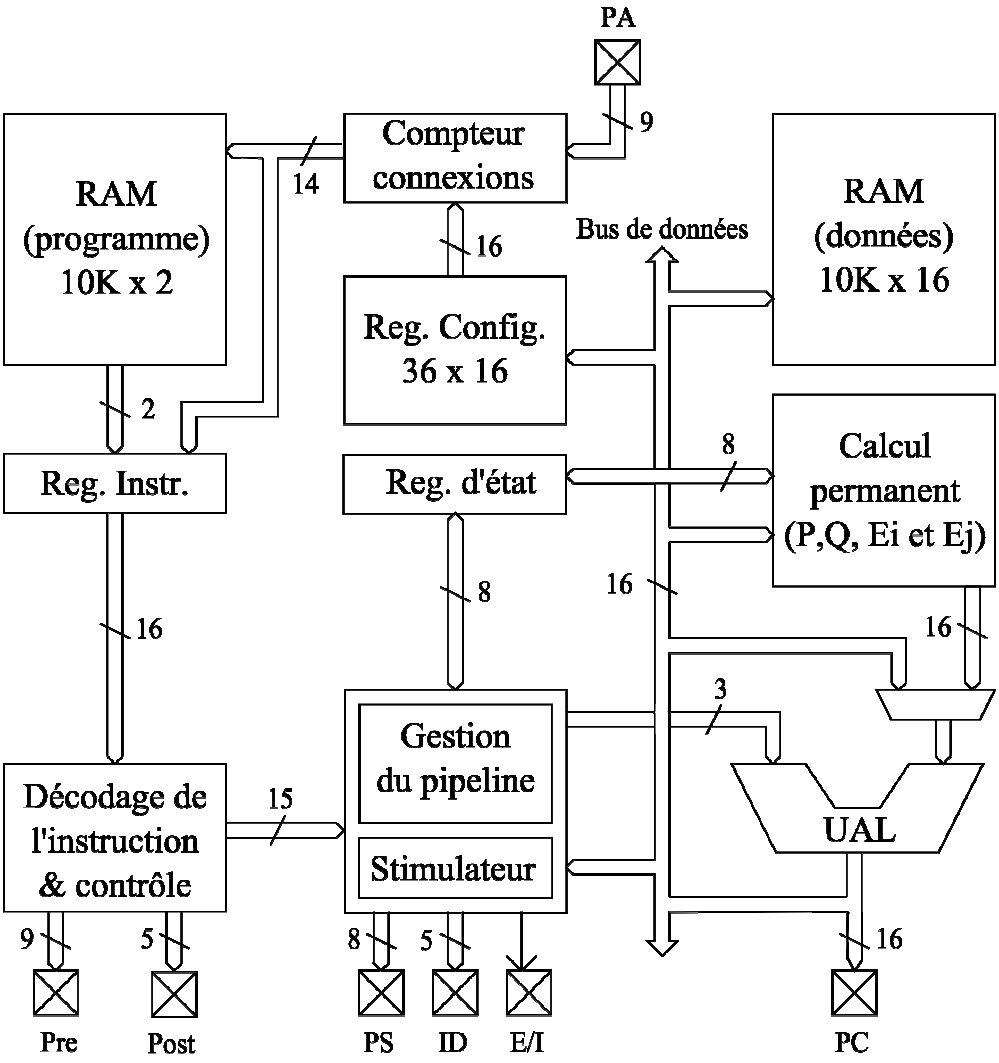



Figure 4.12. Schéma bloc du processeur STDP

Le processeur fonctionne à 100 MHz et est configuré à l'aide d'une séquence particulière de mots réservés. Pour des raisons de simplicité, le protocole de configuration n'est pas présenté ici, il est l'objet de l'annexe B.

Une fois la prochaine instruction est déterminée, elle est stockée avec la valeur du *compteur connexions* dans le registre d'instruction. La valeur du *compteur connexions* détermine les neurones pré et postsynaptique qui construisent la connexion. Le bloc *décodage*

*de l'instruction & contrôle* filtre les connexions valides (connexions existantes) et envoie leurs coordonnées vers le bloc de traitement suivant. A son tour, le bloc suivant distingue entre les connexions plastiques et non plastiques. Celles qui ne sont pas plastiques ne seront pas insérées dans le *pipeline*[1] *du processeur* pour suivre un calcul de plasticité optimisé. Par contre, toutes les connexions, plastiques ou non, passent par le *stimulateur* qui envoie les données d'activation aux multisynapses. Il génère en sortie la valeur du poids de stimulation (PS) codée sur 8 bits, l'identificateur du neurone postsynaptique cible (ID) et la nature de la stimulation (E/I : excitation ou inhibition).

Le jeu d'instructions développé pour ce processeur consiste en des commandes de calcul ou de mises à jour des fonctions de plasticité selon la connectivité du réseau. La matrice de connectivité joue, en quelque sorte, le rôle du programme du processeur. Le calcul se fait à l'aide d'un pipeline à 10 étages. Il permet une utilisation efficace de l'unité arithmétique et logique (UAL), et les accès mémoire. Ainsi, une centaine de demandes de calcul qui durent 10 cycles d'horloge chacune peuvent être compressés en 110 cycles (au lieu de 1000 cycles). Cependant, le pipeline devient moins efficace lorsque les instructions sont espacées dans le temps. Il devient même inutile si les instructions sont régulièrement espacées les unes des autres de plus de 10 cycles d'horloge.

Un bloc de *calcul permanent* est en charge de calculer les fonctions P, Q, $\varepsilon_j$ et $\varepsilon_i$ de l'équation (4.16). Ces fonctions contiennent des exponentielles décroissantes dépendantes du temps, d'où la nécessité d'un calcul permanent. Les états de progression de ces fonctions sont indiqués dans un registre particulier, appelé *registre d'état*. Le bloc *gestion du pipeline* consulte ce registre pour accéder aux valeurs des fonctions ou pour insérer des mises à jour.

L'UAL effectue les calculs nécessaires pour déterminer le nouveau poids, ou poids récemment calculé (PC). Le calcul d'un poids synaptique prend 9 cycles d'horloge au maximum (s'il n'y a pas de pipeline) et 1 cycle au minimum (avec pipeline). Le PC sera réinséré dans la mémoire de données à la place de l'ancien. Il sera également transmis avec les coordonnées de la connexion en question (*Pre*, *Post*) à l'extérieur, où il sera considéré comme un résultat de simulation.

Dans le chapitre 3, on a montré que le calcul des fonctions P, Q, $\varepsilon_j$ et $\varepsilon_i$ donne de meilleures performances en utilisant l'approche d'implémentation orientée-cellule. Cependant, le nombre de cellules a beaucoup augmenté dans la version de PAX3 (500 cellules) par rapport à la version PAX2 (25 cellules). Les ressources matérielles ne permettent pas d'implémenter les fonctions de plasticité pour chaque cellule. Une solution faisant intervenir un multiplexage temporel s'avère nécessaire.

La solution consiste à allouer des créneaux de temps identiques pour rafraîchir en permanence les valeurs des exponentielles décroissantes des 500 neurones à partir d'une seule implémentation des blocs des fonctions. D'après l'algorithme de distribution du calcul de la plasticité décrit par la figure 4.10, on aura besoin de calculer les exponentielles des fonctions P et $\varepsilon_j$ pour toutes les cellules du réseau, alors qu'on n'aura besoin de calculer les exponentielles des fonctions Q et $\varepsilon_i$ que pour les neurones locaux.

Pour optimiser le processus de rafraîchissement permanent, le calcul est accéléré par un deuxième pipeline interne aux blocs des fonctions. Ce qui permet d'effectuer un rafraîchissement à chaque cycle d'horloge. Ainsi, on aura besoin d'environ 500 cycles pour rafraichir les valeurs de la fonction P et $\varepsilon_j$. Ceci impose une valeur limite minimale de la

[1] Le pipeline d'un processeur fait référence à l'élément dans lequel l'exécution des instructions est découpée en étages pour accélérer les calculs. Il peut être vu comme un module permettant d'exploiter au maximum les ressources d'un processeur.

constante de temps $\tau_p$ qui conduit à un rafraîchissement périodique, de période supérieure à 500 cycles d'horloge ($P_0 = \tau_p/2^n > 500$ cycles, où n est le nombre de bits utilisés pour le codage).

*Performances du processeur*

L'implémentation du processeur STDP a été réalisée sur un FPGA *Xilinx Spartan 3 XC3S1500FG456*, 32 multiplieurs dédiés et 32 blocs RAM de 18 kbits chacun. Le tableau 4.1 fournit les pourcentages de l'utilisation des ressources du FPGA après les phases de placement et routage du code VHDL du processeur.

Tableau 4.1. Pourcentage d'utilisation des ressources matérielles du processeur STDP après implémentation sur un FPGA *Spartan 3*.

| | **Fréquence FPGA** | **Bloc RAM** | **Multiplieurs** | **LUTs** |
|---|---|---|---|---|
| **Implémentation multiplexée** | 100 MHz | 14 (43%) | 6 (18%) | 2516 (9%) |

En comparant ces résultats avec ceux obtenus pour la version de l'implémentation de la plasticité dans le système PAX2, on observe une augmentation de l'utilisation des blocs RAM. Cette augmentation s'explique par l'augmentation du nombre de neurones et l'utilisation de la technique de multiplexage temporel dans le calcul des fonctions synaptiques. L'utilisation des autres ressources du FPGA reste relativement faible.

Généralement, l'utilisation de la technique du multiplexage temporel induit des temps de traitement supplémentaires dans le calcul. Parmi les solutions qui permettent de lutter contre ces délais, c'est l'utilisation du pipeline qui a été retenue. Cette version du processeur emploie un premier pipeline dans le jeu d'instructions et un deuxième entre les blocs de calcul permanent. Le pipeline utilisé dans le calcul permanent est toujours efficace, alors que celui du jeu d'instructions dépend de l'espacement entre les instructions, comme précédemment expliqué. Ceci conduit à une différence dans le temps de traitement du processeur. On distingue deux cas limites : le pipeline est exploité au maximum (pipeline efficace), et un pipeline jamais exploité (pipeline non efficace).

La figure 4.13 présente l'évolution du temps de traitement du processeur en fonction du nombre de connexions dans le réseau. Le processeur est supposé dans un régime de fonctionnement extrême, ou chaque connexion est plastique et nécessite un calcul. Les deux courbes présentent respectivement le temps de traitement obtenu avec un pipeline efficace et un pipeline non efficace. L'espace entre ces deux courbes présente des valeurs intermédiaires possibles du temps de traitement avec des pipelines partiellement efficaces.

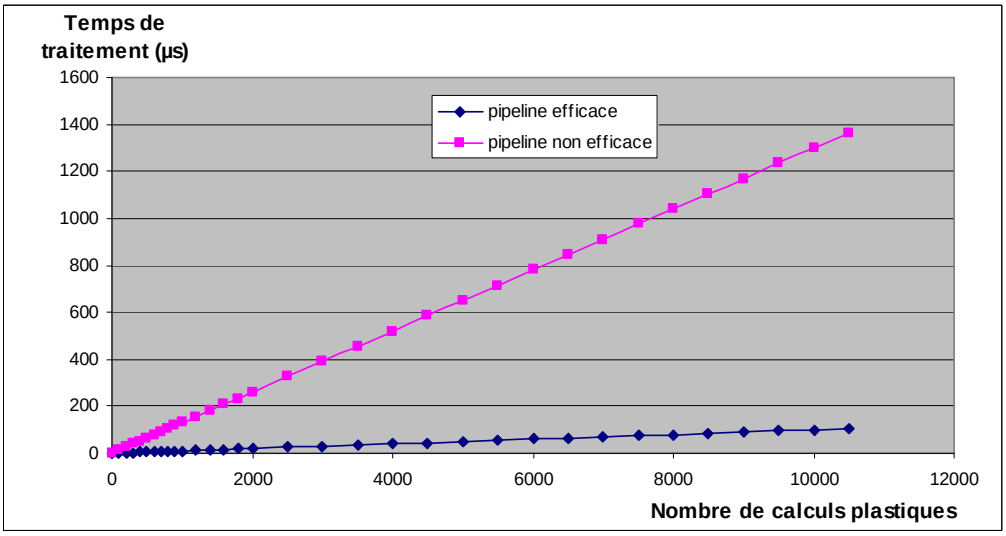



Figure 4.13. Evolution du temps de traitement du processeur STDP en fonction du nombre de connexions et de l'efficacité du pipeline.

Rappelons que la contrainte temps réel appliquée au calcul de la plasticité consiste à dire que le calcul de la plasticité doit se faire en temps inférieur à celui qui sépare l'instant où un neurone génère un potentiel d'action et l'instant où il se prête à générer un deuxième.

Le cahier de charge énonce un taux maximal d'oscillation des neurones de 100 évènements/s. Dans les conditions du pire cas, le processeur de plasticité doit faire face aux 10400 calculs des poids synaptiques chaque 10 ms (le temps qui sépare deux évènements successifs). La figure 4.13 montre que le temps de traitement de 10400 connexions dans le cas extrême vaut 1.36 ms avec un pipeline non efficace, ce qui est largement inférieur à 10 ms. La contrainte temps réel du calcul de la plasticité est donc satisfaite. Il est même possible d'augmenter les taux d'oscillation de chaque neurone pour atteindre les 735 évènements/s.

*Validation du calcul de la STDP*

Afin de valider le fonctionnement du calcul de la plasticité, on a développé un simulateur logiciel pour simuler le processeur STDP. Le simulateur logiciel réalise les opérations arithmétiques à virgule flottante, ce qui donne plus de précision au calcul. On fournit la même configuration et les mêmes activités des neurones au simulateur logiciel (les potentiels d'action et de leurs heures d'apparition). Ensuite, on compare les deux résultats pour conclure sur la validité du calcul matériel.

Les neurones analogiques n'interviennent pas dans la simulation. Leur activité est simulée par un générateur de potentiels d'action permettant de mieux maîtriser la réponse du système. Son but est d'imposer une activité aux neurones dont le résultat du calcul de plasticité est facilement prévisible.

La figure 4.14 illustre un premier test qui consiste à connecter mutuellement deux neurones et de les faire osciller de telle sorte que l'évolution des poids synaptiques se fait dans un seul sens (potentiation ou dépression). Les deux neurones oscillent à la même fréquence avec un décalage constant des phases. Le comportement prévu est donc une augmentation progressive dans les poids synaptiques d'une connexion (une série de LTP) et une diminution progressive des poids de la deuxième connexion (une série de LTD).

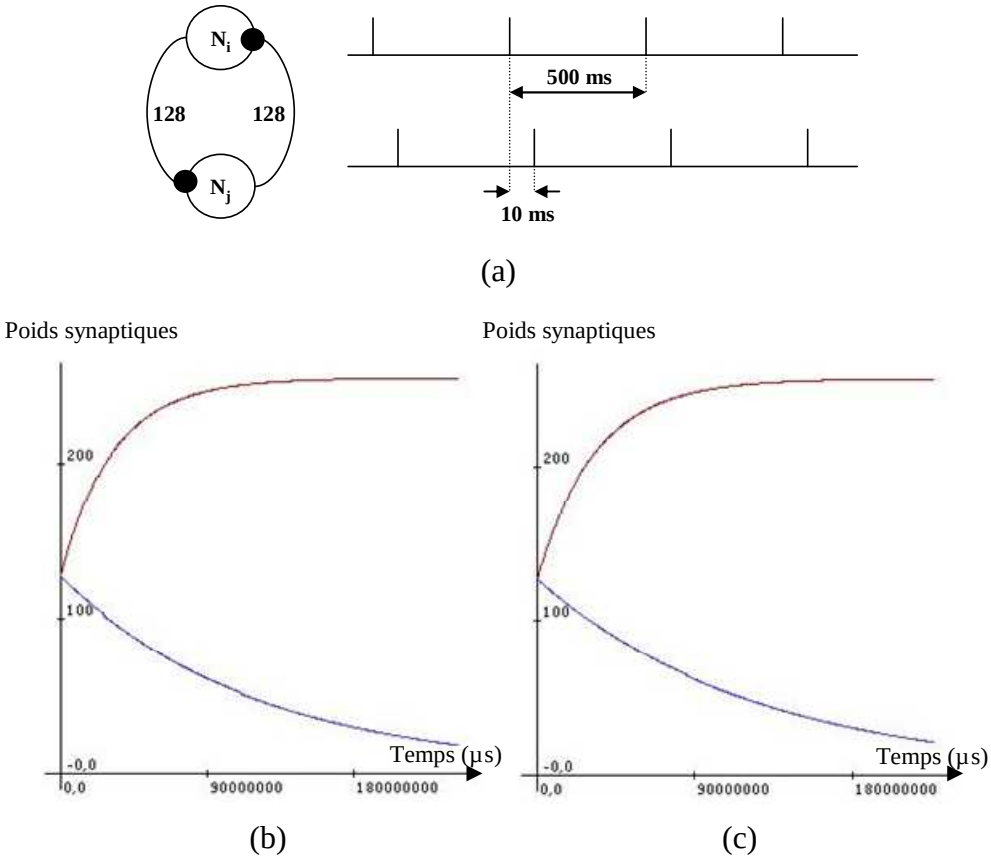
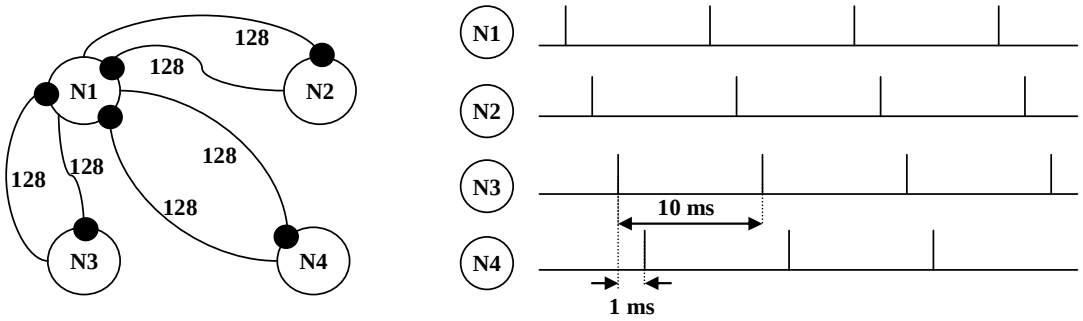



Figure 4.14. Validation du calcul de la LTP et la LTD par le processeur STDP : application des règles de l'équation (1.12). Les paramètres de l'équation valent : $\tau_p$ = 14,8 ms, $\tau_q$ = 33,8 ms, $\tau_j$ = 28 ms, $\tau_i$ = 88 ms, $A_p$ = 0,1, $A_q$ = 0,05, $w_{LTP}$ = 255 et $w_{LTD}$ = 0. (a) La spécification du réseau consiste en deux neurones mutuellement connectés qui oscillent à la même fréquence à des phases différentes. (b) L'évolution des poids synaptiques obtenue à partir d'une simulation matérielle. (c) L'évolution des poids synaptiques obtenue à partir d'une simulation logicielle (µs).

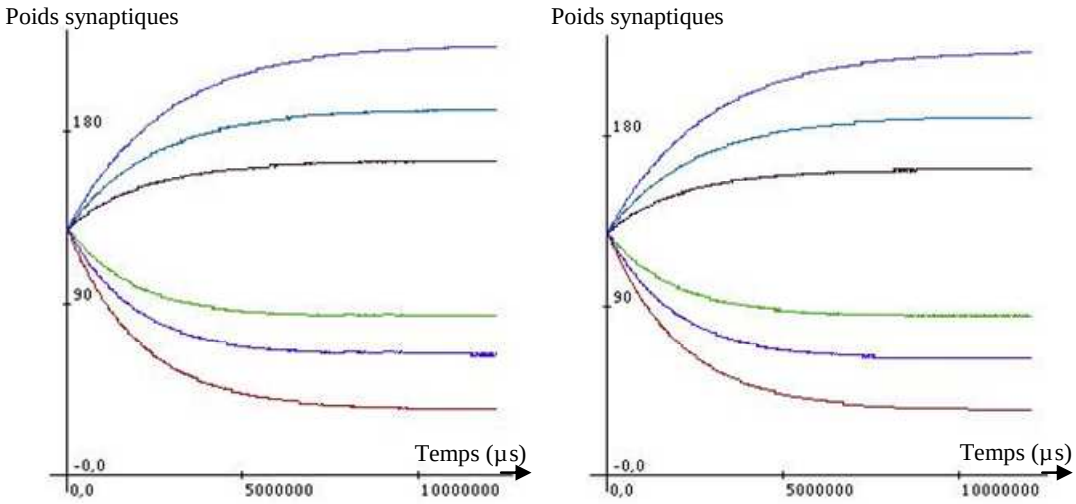
Le premier graphe représente le résultat de la simulation matérielle. Le résultat est conforme aux prédictions : une connexion qui augmente constamment, alors que la deuxième diminue. L'évolution tend vers les valeurs limites sans jamais les atteindre mettant en évidence l'effet des fonctions de suppression, $\varepsilon_j$ et $\varepsilon_i$. En plus, des effets de saturation apparaissent de plus en plus en s'approchant des valeurs limites définissant la saturation *soft bound* de la synapse.

Le deuxième graphe représente le résultat de la simulation logicielle. Le résultat est très proche de celui obtenu en simulation matérielle. Les comportements des deux simulations sont similaires, permettant de valider ainsi le calcul de la potentiation et de la dépression par le processeur STDP. Cependant, il existe des différences dans les valeurs finales. L'analyse de ces différences fera l'objet du paragraphe suivant.

La figure 4.15 illustre une simulation similaire mais en utilisant un réseau composé de 4 neurones et 6 connexions. Tous les neurones sont mutuellement connectés au premier neurone *N1*. La fréquence d'oscillation est la même pour tous les neurones. Cependant, les phases varient d'un neurone à un autre avec un écart constant, comme le montre la figure 4.15 (a). On prévoit que le résultat sera composé de 6 évolutions distinctes des poids synaptiques : 3 potentiations et 3 dépressions.




(a)




(b) (c)

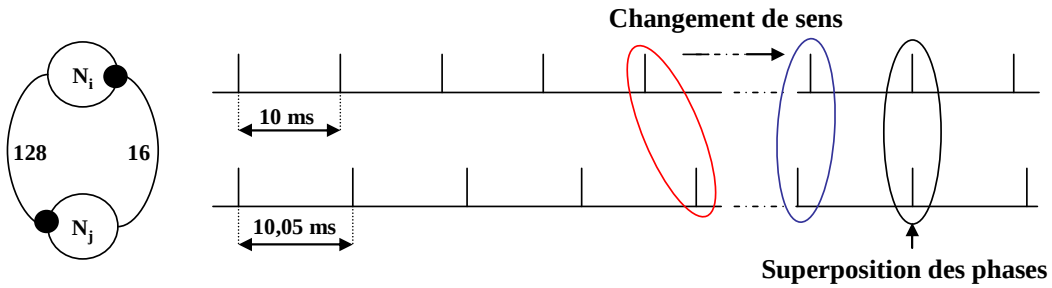
Figure 4.15. Simulations d'un réseau composé de 3 neurones et 6 connexions : les paramètres de l'équation de la plasticité sont identiques pour la LTP et la LTD : $\tau_p = \tau_q = 14{,}8$ ms, $\tau_j = 28$ ms, $\tau_i = 88$ ms, $A_p = A_q = 0{,}1$, $w_{LTP} = 255$ et $w_{LTD} = 0$. (a) La spécification du réseau est réalisée de telle sorte que tous les neurones sont connectés dans les deux sens au neurone *N1*. (b) Résultat de la simulation matérielle : les poids convergent vers 6 valeurs finales différentes. (c) La simulation logicielle donne des valeurs très proches.

Les résultats obtenus confirment les prédictions de départ. Dans cette expérimentation, nous avons utilisé des constantes de temps et des amplitudes identiques pour LTP et LTD ($\tau_p = \tau_q = 14{,}8$ ms et $A_p = A_q = 0{,}1$) et des poids initiaux égaux à 128, ce qui explique l'allure symétrique de l'évolution des poids des connexions.

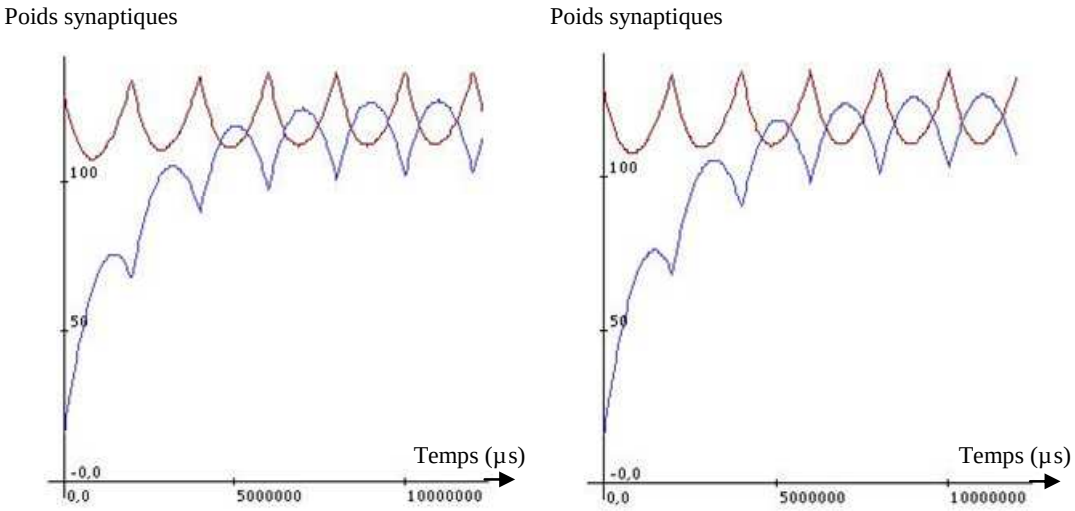
Encore une fois, le calcul logiciel valide les résultats obtenus à partir de la simulation matérielle. Cette expérimentation a permis de valider la configuration du processeur et la dépendance des connexions à la différence des phases entre les potentiels d'action pré et postsynaptiques.

La figure 4.16 illustre un autre type d'expérimentation pour la validation des fonctionnalités du calcul de la plasticité. On reprend ici un réseau de deux neurones mutuellement connectés. Les poids initiaux sont différents. L'expérience consiste à faire varier les phases des évènements progressivement et régulièrement. Pour le faire, on a fixé des fréquences d'oscillation légèrement différentes pour les deux neurones (100 Hz et 99,9 Hz). Le scénario attendu consiste en l'augmentation progressive de la différence de phases des deux neurones jusqu'à avoir un changement de sens dans l'évolution des poids en passant d'une potentiation à une dépression ou le contraire. Il y aura aussi une coïncidence

entre des évènements traduisant une superposition des phases. Chaque coïncidence marque le début d'un nouveau cycle de calcul identique au précédent.




(a)




(b) (c)

Figure 4.16. Simulation d'un réseau de deux neurones et deux connexions : les fréquences d'oscillation sont légèrement différentes pour créer une périodicité dans les changements des phases. Les paramètres du calcul de la plasticité sont les même que l'expérimentation illustrée par la figure 4.14. (a) Spécification du réseau : les poids initiaux ont des valeurs différentes (16 et 128). Les changements de sens dans le groupement des paires de potentiels d'action créent, à un certain moment, des inversions de signes dans la différence des phases. Quand cette dernière est nulle, on parle alors de superposition de phases. (b) Simulation matérielle conforme aux résultats attendus : les deux connexions évoluent d'une façon antagoniste en présentant des changements de sens périodiques. (c) La simulation logicielle : un comportement similaire au calcul matériel.

Le poids de la première connexion démarre de la valeur 128 pour décroître puis croître périodiquement. La phase de descente dure moins longtemps que la phase de remontée. Ceci est dû à la différence des cinétiques de LTP et LTD utilisées lors de la configuration des modules de calcul ($\tau_p$ = 14,8 ms et $\tau_q$ = 33,8 ms). Les deux phases combinées ensemble durent 2 s : c'est le résultat d'une augmentation de la différence de phases de 0,05 ms à chaque oscillation.

Les pointes périodiques des courbes marquent la fin des cycles de 2 s. Les potentiels d'actions sont supposés simultanés à cet instant. Or, le modèle de plasticité présente une discontinuité dans le cas où l'intervalle de temps entre les évènements est nul. Il n'est pas déterminé si le calcul procédera selon une potentiation ou une dépression. Ce problème a été évité lors de l'implémentation matérielle, où il n'existe pas des évènements simultanés : le

traitement séquentiel des potentiels d'action impose que les évènements soient traités l'un après l'autre.

Le poids de la deuxième connexion démarre de la valeur 16 et commence à osciller de la même façon que la première connexion. Cependant, les phases de remontée et de décente sont inversées. C'est le résultat d'une inversion dans les neurones pré et postsynaptiques. De plus, les premières phases de remontée effectuent des grandes potentiations contre des petites dépressions dans l'évolution des poids qui tend à s'équilibrer par la suite. L'explication vient du fait que les facteurs de saturation sont calculés en fonction de l'ancien poids synaptique : plus l'ancien poids est faible plus la variation de LTP est grande et la variation de LTD est faible, et *vice-versa*.

La version logicielle confirme le comportement obtenu par le processeur matériel. D'autres expérimentations réalisées avec un nombre plus important de neurones et de connexions ont également validé le bon fonctionnement du processeur STDP. A un comportement conforme à la théorie, il reste à étudier la précision du calcul de la plasticité.

*Différence des valeurs finales à activités égales*

En examinant la différence entre le calcul matériel et le calcul logiciel, on trouve des écarts plus au moins grands entre les résultats de calcul. En fonction du temps de simulation et de l'évolution des poids synaptiques des connexions, cet écart varie positivement ou négativement.

On se propose alors de mettre en évidence les sources de cet écart et de déterminer son impact sur une simulation. On procède de la même façon que précédemment : on effectue des simulations logicielles et matérielles à activités des neurones égales. Par la suite, on trace la différence entre les deux résultats obtenus.

La figure 4.17 fournit la valeur absolue de la différence entre les poids synaptiques d'une connexion en fonction du temps. Dotés d'une évolution monotone, les poids synaptiques augmentent constamment durant toute la simulation. Le comportement est le même dans les deux simulations. Cependant, les valeurs finales des poids sont différentes (figure 1.17 (c)).

La différence augmente en fonction du temps de simulation et s'atténue progressivement pour se stabiliser à la fin à la valeur de 7. L'évolution de la différence est reliée à l'évolution des poids synaptiques. Plus la variation des poids est grande plus la différence augmente rapidement, ce qui est le cas du début de la simulation. Vers la fin de la simulation, plus la variation des poids sature, plus la différence se stabilise à son ancienne valeur.

Dans la phase de saturation, les poids synaptiques changent peu ou jamais, ce qui les rend moins sensibles à la variation des valeurs du calcul et explique la stabilisation de la différence autour de la valeur 7.

Cette différence des valeurs finales à activités égales s'explique par le fait que le calcul logiciel se fait par rapport à une activité des neurones simulés matériellement. Etant donné que la réponse des neurones est propre à la simulation matérielle, c'est la précision de calcul matérielle qui sera prise en compte. Par conséquent, la version logicielle qui utilise le résultat de l'activité des neurones matériels calculera des valeurs de poids synaptiques différentes. Ces différences s'accumulent au cours du temps dans les calculs des poids synaptiques qui suivent.

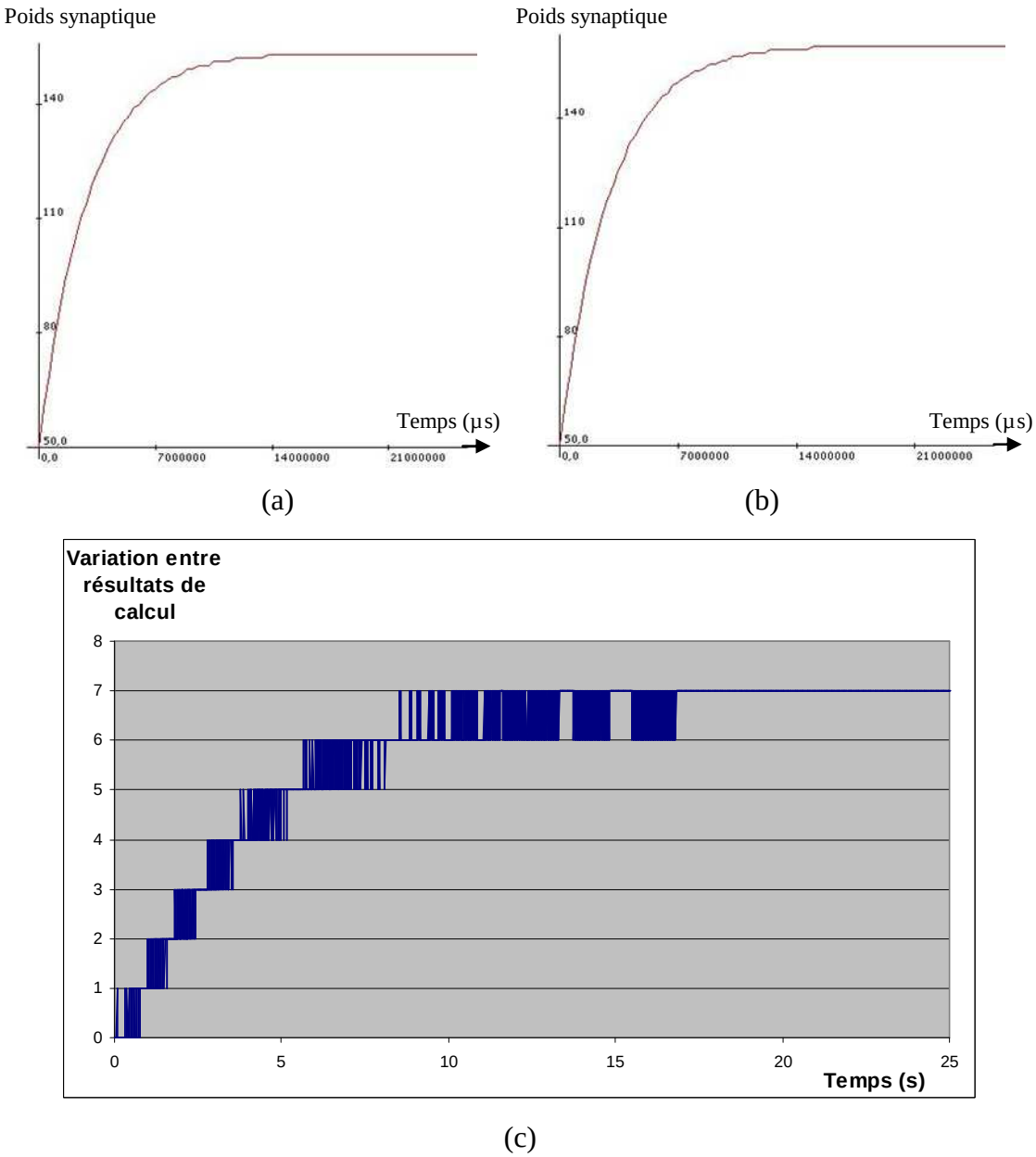



Figure 4.17. Quantification de l'erreur d'une connexion référence. (a) Résultat issu d'une simulation logicielle à virgule flottante. (b) Résultat issu d'une simulation matérielle à virgule fixe. (c) Valeur absolue de la différence entre les deux courbes.

La différence finale dépend de plusieurs paramètres parmi lesquels le temps de simulation, la cinétique des calculs de la plasticité ($\tau_p$ et $\tau_q$), les facteurs de saturation ($w_{LTP}$ et $w_{LTD}$) et les fonctions de suppression ($\tau_j$ et $\tau_i$), mais aussi de la précision de calcul des poids synaptiques. Il ne devient donc pas possible de la cerner dans une marge de valeurs et estimer son impact sur le résultat final. Par contre, il est possible de la diminuer ou même de la neutraliser par les actions antagonistes du calcul de la plasticité, à savoir la LTP et la LTD.

L'alternance des actions de LTP et LTD compense les différences de calcul : le poids synaptique augmente quand il y a LTP (valeur positive), et diminue lorsqu'il y a LTD (valeur négative). L'expérience évidente qui permet de vérifier cette caractéristique est la détermination de la valeur absolue de la différence des résultats finaux dans une connexion à

évolution périodique. La figure 4.18 présente une connexion reliant deux neurones de fréquences 100 et 99.9 Hz, ce qui crée un changement de phases périodique de période 10 s.

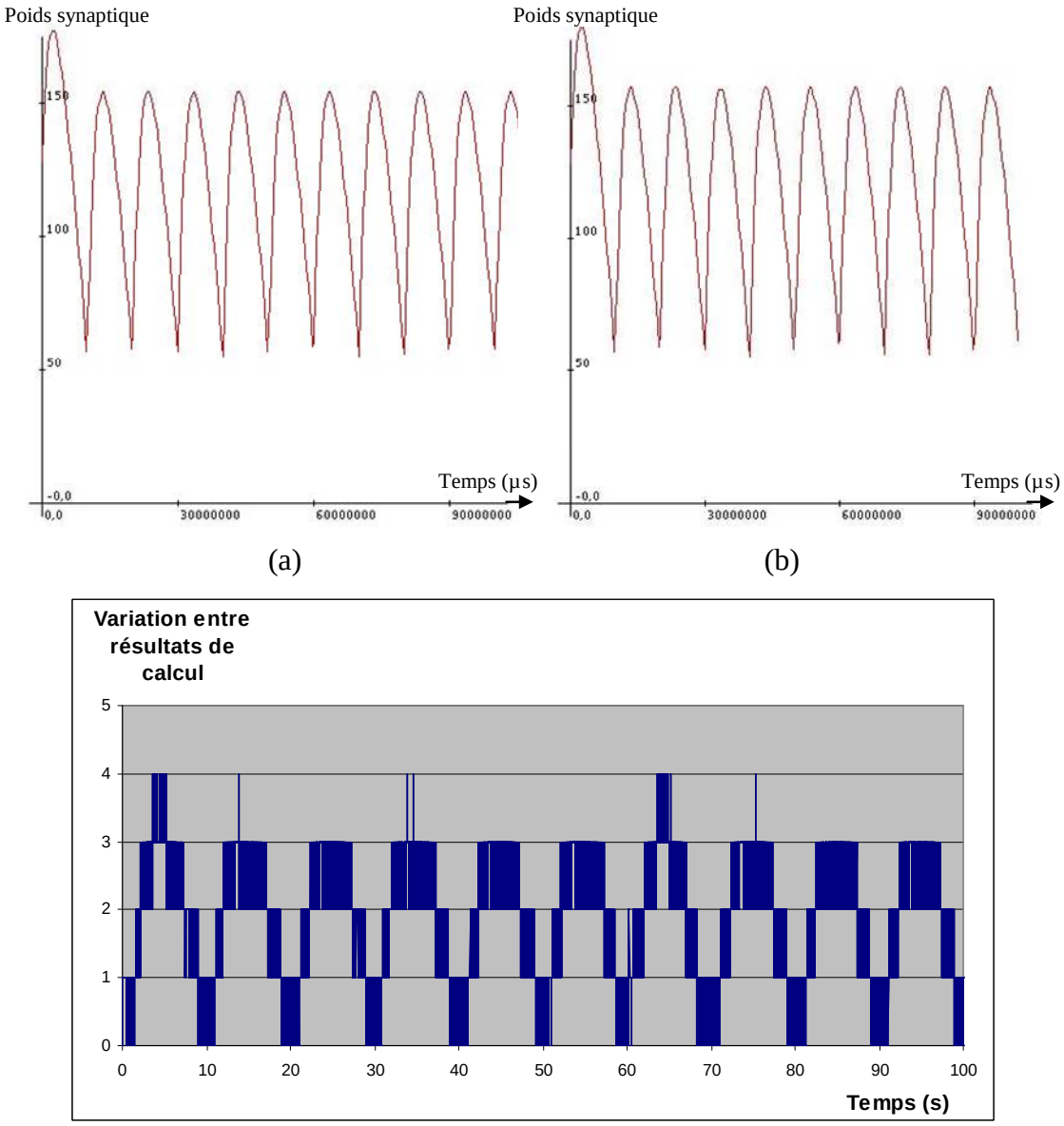



Figure 4.18. Mise en évidence de l'erreur accumulée. (a) Résultat de la simulation matérielle d'une connexion à évolution périodique. (b) Résultat de la simulation logicielle. (c) Valeur absolue de la variation de l'erreur entre le calcul logiciel et le calcul matériel.

La différence entre les poids synaptiques issus d'un calcul logiciel et un calcul matériel suit la périodicité de la courbe. Lors de la phase de remontée (une série de LTPs), la valeur absolue de la différence augmente progressivement. Au contraire, pendant la phase de descente (une série de LTDs), la différence diminue progressivement pour revenir au niveau de départ. La même évolution se répète périodiquement chaque 10 s. Cette période correspond à celle de l'évolution de la connexion.

De plus, la variation de la différence suit la cinétique des phases de la remontée et de la descente de la courbe : la phase de remontée dure moins que la phase de descente. L'incertitude accumulée est donc non seulement dépendante de l'alternance de LTP et de LTD mais aussi de leur taux d'accroissement.

En résumé, l'écart entre les valeurs finales des simulations matérielles et logicielles est fonction, entre autres, de la précision de calcul des poids synaptiques. Cependant, cette différence de précision peut être diminuée ou annulée par l'action de la convergence des poids synaptiques vers des valeurs stables et l'alternance des fonctions antagonistes LTP et LTD du modèle de plasticité. Sachant que dans le cas d'une simulation matérielle, les circuits analogiques ne fournissent pas des potentiels d'action avec des fréquences parfaitement synchronisées ou identiques, ce qui renforce l'occurrence des actions antagonistes durant la simulation. L'erreur induite par la précision de calcul des poids synaptiques est donc minimisée. Une étude plus approfondie de la neutralisation de cette erreur pourrait se faire lorsque le réglage des circuits de neurones sera achevé.

## 4. L'intégration globale

Les subtilités liées au fonctionnement global d'un système émergent lors de l'intégration de ses composants constitutifs. Le système PAX3 repose sur l'intégration de plusieurs exemplaires de cartes fille et d'une carte mère. Cette intégration ajoute des traitements et des contraintes supplémentaires afin de réaliser des simulations correctes.

### 4.1 La nécessité d'une classification des évènements

Le protocole de communication garantit l'heure d'arrivée des évènements à leurs cibles, mais il ne garantit pas leur ordre d'arrivée. Ainsi, un évènement généré à un temps $t_x$ peut arriver à la destination après l'arrivée d'un deuxième généré à un temps $t_y > t_x$. Sachant que le calcul de la plasticité (STDP) repose sur les anciennes valeurs des poids synaptiques, un mauvais ordre peut complètement altérer les résultats de simulation.

Il devient donc nécessaire d'ajouter un traitement de classification des évènements en amont du premier calcul à réaliser. De plus, le traitement doit être rapide pour conserver la contrainte temps réel du système global.

La solution que l'on propose tire profit de la garantie de l'heure limite d'arrivée des évènements à leurs destinations (D). Prenons l'exemple de la version actuelle de PAX3 où $D = 5\mu s$, les évènements sont étiquetés par l'heure d'apparition (absolue) et la résolution temporelle est la microseconde. Les évènements qui arrivent à une carte donnée sont classifiés selon leur heure d'apparition. Etant donné que des évènements peuvent arriver à leur destination dans un intervalle de temps allant de 1 à 5 microsecondes, on prévoit 5 heures d'apparition différentes. Les évènements sont donc séparés en 5 lots ; un lot représente une heure d'apparition. Chaque lot est stocké dans une FIFO spécifique, comme le montre la figure 4.19.

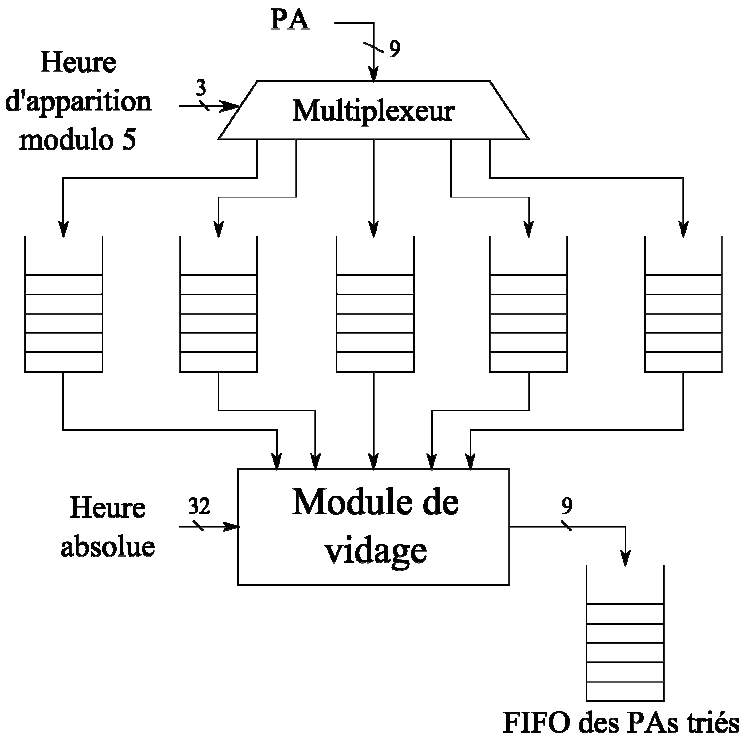



Figure 4.19. Schéma bloc de la procédure de classement des évènements (PA).

Les FIFOs sont vidées à tour de rôle par l'ordre croissant. Le temps de début de vidage de chaque FIFO correspond à l'heure d'apparition qu'elle présente, à laquelle on additionne la période D. L'ajout de D garantit que tous les évènements d'une heure d'apparition sont déjà arrivés et classés.

Les cycles de classification des évènements et de vidage des FIFOs se répètent jusqu'à la fin de la simulation. Une sixième FIFO accueille les évènements classés. Ces derniers sont donc triés par ordre d'apparition, et ils serviront comme entrée pour le calcul de la plasticité et la stimulation des neurones cibles.

En conclusion, l'opération de classification a ajouté un temps supplémentaire à la date d'arrivée des évènements. Ce temps est fixe et vaut D. Par conséquent, l'heure limite d'arrivée des évènements est multipliée par deux (2D). Ceci conduit à une valeur de 10 µs dans la version actuelle de PAX3. La contrainte temps réel reste cependant satisfaite.

### 4.2 La récupération des données de simulation

Les résultats de simulation permettent de tracer l'évolution de l'activité des neurones et de la plasticité du réseau. A chaque fois qu'un neurone génère un potentiel d'action, ce dernier est archivé avec son heure d'apparition. De même, à chaque fois qu'il y a une mise à jour d'un poids synaptique, il est archivé avec les coordonnées de sa connexion.

Toutes les données de simulations sont envoyées en continu à une machine hôte. La carte mère scrute le bus de communication, récupère les messages et les transforme en des trames USB pour les transmettre à la fin à la machine hôte. Le débit requis pour la liaison USB est fonction de la taille du réseau, de la connectivité et de l'activité des neurones. Pour déterminer le débit maximal requis, on doit se placer dans les conditions du pire cas. La figure 4.20 fournit l'évolution du débit maximal en fonction de la taille du réseau.

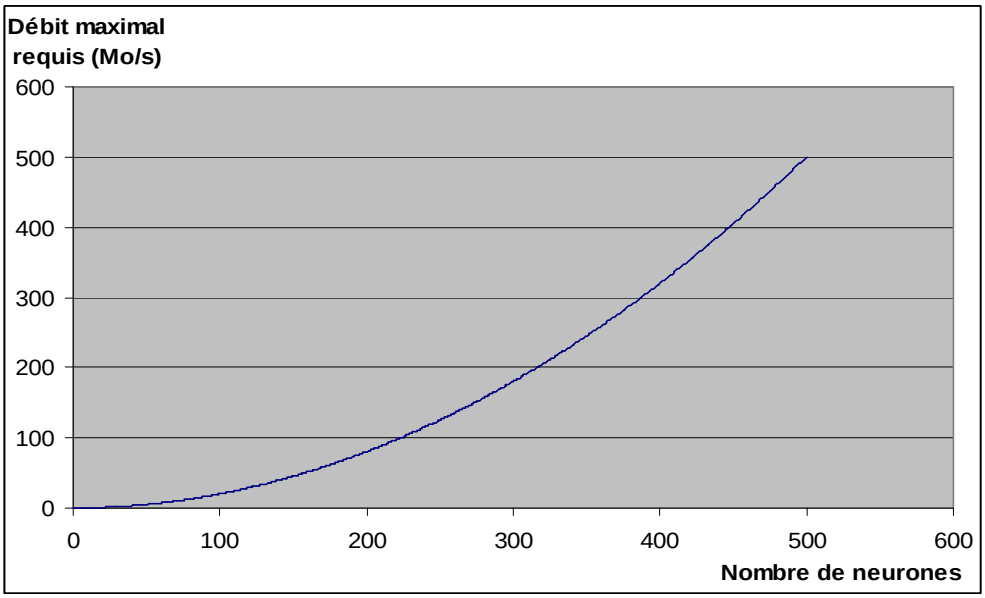



Figure 4.20. Débit requis pour véhiculer les résultats de simulation dans les conditions du pire cas.

Dans la version actuelle de PAX3, la liaison USB permet un débit maximal de 25 Mo/s. Ceci limite la taille de réseau à 112 neurones fonctionnant dans le pire cas (voir figure 4.20). Il n'est donc pas possible de garantir la récupération de tous les résultats de simulation pour un réseau plus grand. Cette limite sera comblée dans la prochaine version de la carte mère qui contiendra des lecteurs cartes SD[1] et des liaisons USB et Ethernet haut débit pour permettre de récupérer un nombre plus important de données de simulation.

### 4.3 La stimulation synaptique

La stimulation synaptique se fait par l'intermédiaire de la multisynapse. Cette dernière somme les valeurs des poids synaptiques des connexions afférentes et les transforme en une impulsion numérique dont la largeur détermine l'intensité du courant synaptique. La largeur de l'impulsion ne doit pas excéder 60 µs.

A ce niveau, on peut aussi imaginer un pire cas : des stimulations présynaptiques maximales arrivant consécutivement sur un même neurone dans un réseau de neurones complètement connecté (*all-to-all*). La contribution de tous les neurones présynaptiques du réseau avec des poids maximaux conduit à la plus grande somme des poids synaptiques, qui conduit, à son tour, à l'impulsion la plus large.

Pour bénéficier du calcul précis des poids synaptiques, il faut penser à représenter les 8 bits du codage de chaque poids par une proportion de la largeur de l'impulsion. Ceci revient à dire que si le composant qui implémente les multisynapses est cadencé avec une horloge de 100 MHz, on devrait attribuer au moins 10 ns (un cycle d'horloge) à la valeur unitaire du poids (poids = 1). Dans le cas où le poids est maximal (poids = 256), la durée devient 256*10 ns, soit 2,56 µs à compter sur la largeur de l'impulsion.

Le pire cas suppose que la durée de 2,56 µs est multipliée par le nombre de neurones présynaptiques. Autrement dit, si on a 100 neurones présynaptiques, on pourrait générer une impulsion de largeur 256 µs, ce qui dépasse nettement la largeur maximale autorisée.

[1] La carte SD (SD étant le sigle de l'anglais *Secure Digital*) est une carte mémoire amovible de stockage de données numériques. Elle permet de stocker quelques giga octets à une vitesse qui peut atteindre les 10 Mo/s.

Dans la version actuelle du système PAX3, il est possible de faire tourner des simulations d'un réseau constitué de 112 neurones. Le réglage des types de neurones sera fait selon les proportions biologiques : 80% de neurones excitateurs contre 20% de neurones inhibiteurs, ce qui donne 89 neurones excitateurs contre 23 inhibiteurs. Pour chaque neurone, il existe deux multisynapses, une qui excite (injection de courant positif) et une deuxième qui inhibe (inversion du signe du courant). Les neurones présynaptiques excitateurs seront reliés aux multisynapses excitatrices, et les inhibiteurs aux multisynapses inhibitrices (voir figure 4.21).

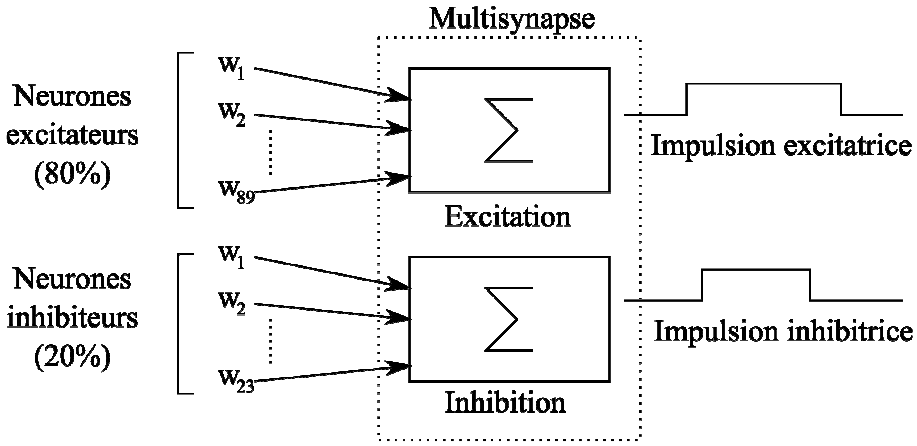



Figure 4.21. Organisation de la multisynapse dans le système PAX3

Par conséquent, l'impulsion inhibitrice peut atteindre les 58,9 µs de largeur alors que l'impulsion excitatrice peut dépasser la largeur maximale autorisée et atteindre les 222,5µs dans le pire cas. Pour remédier à ce problème, il faut réduire la largeur de l'impulsion au détriment de la précision. Deux méthodes sont possibles :

- Négliger les deux premiers bits de poids faible : seuls les 6 bits du poids fort du poids synaptique seront pris en compte. Ceci permet de générer une impulsion de largeur maximale 56,9 µs. Les stimulations qui se font avec des poids synaptiques compris entre 1 et 3 seront omises.
- Sommer les poids jusqu'à plafonner la largeur maximale autorisée : on garde la précision des 8 bits des poids synaptiques, par contre, on borne leur somme de telle sorte que l'impulsion ne dépasse pas sa valeur maximale. Cette méthode peut écarter des poids synaptiques entiers de la contribution à la stimulation synaptique. En termes du pire cas, seulement 24 neurones sur 89 sont considérés dans une stimulation.

La première méthode est souhaitable lorsque les poids synaptiques n'atteignent pas les valeurs basses. Il est possible de configurer l'algorithme de plasticité pour qu'il sature vers une valeur minimale personnalisée des poids ($W_{LTD}$).

L'avantage de la deuxième méthode est quelle conserve la précision du calcul de la plasticité et sensibilité de la stimulation. Elle est souhaitable pour des réseaux faiblement connectés ou à faible nombre de neurones.

## 5 Conclusion

Dans ce chapitre, on a effectué une étude complète pour la réalisation d'un système multicartes/multicircuits qui vise à simuler des réseaux de neurones de taille relativement

grande. Dans une première partie, un protocole de communication garantissant l'aspect temps-réel lors du dialogue inter-cartes a été présenté. La topologie d'accès anneau à jeton du protocole a permis une allocation équilibrée de la bande passante du bus de communication. Exploitée par une méthode intégrative d'allocation de la bande passante, on a pu garantir la satisfaction de la contrainte temps-réel relative à la communication.

D'un autre côté, on a présenté une étude pour la réalisation d'une architecture distribuée pour la gestion du réseau. Ce qui nous a conduits à modifier l'algorithme de la plasticité (initialement conçu pour fonctionner dans une architecture centralisée) pour distribuer le calcul sur plusieurs cartes. On s'est intéressé également à l'optimisation du calcul de la plasticité pour satisfaire la contrainte temporelle qui lui est relative. Un processeur dédié au calcul de la STDP des connexions synaptiques a été, par conséquent, conçu est réalisé. Les performances de ce processeur satisfont le calcul de la plasticité dans les conditions du pire avec un respect de la contrainte temps-réel.

Les modules de communication et de calcul de plasticité ont été rassemblés pour valider le fonctionnement global du système. Une série de tests ont été réalisés sur le système mettant en évidence le succès de son implémentation. Le système reste d'une nature prototypique et nécessite plus de maturité technologique pour qu'il soit capable d'intégrer un nombre plus grand de neurone. Néanmoins, les méthodes et les techniques utilisées constituent une preuve de la faisabilité d'un simulateur biologiquement réaliste en silicium.

# Conclusion générale

Cette thèse a pour but d'établir un outil de modélisation neuromorphique pour les neurosciences. Son objectif est d'aider à la compréhension des mécanismes neuronaux qui régissent les réseaux de neurones biologiques. A cette fin, une étude de développement des plateformes de simulation matérielles, dites biologiquement réalistes, a été présentée.

Dans cette conclusion générale, on fera le point sur le cadre scientifique et international des travaux de cette thèse. On fera, par la suite, le bilan des travaux effectués et des systèmes réalisés pour terminer par les perspectives qu'ouvrent ces réalisations.

**Cadre Européen**

Ces travaux se sont effectués dans le cadre du projet européen FACETS (FP6-IST-FETPI-2004-15879) (2005-2010). La contribution des projets européens est très importante, puisqu'elle apporte des ressources humaines et matérielles, et crée un cadre pluridisciplinaire avec la présence des modélisateurs et les biologistes. Les biologistes fournissent des protocoles et résultats expérimentaux mettant en évidence les mécanismes neuronaux. Les modélisateurs tirent les principales caractéristiques de ces résultats et les transforment en équations mathématiques exploitables par les concepteurs des circuits et systèmes neuromorphiques.

En parallèle avec l'équipe de K. Meier de l'université de *Heidelberg*, notre équipe intervient au niveau de la conception, la réalisation et l'utilisation des systèmes neuromorphiques. Deux approches de conception sont utilisées. L'équipe de *Heidelberg* exploite la capacité d'intégration des circuits VLSI pour simuler en accélérée des grands réseaux de neurones. Ils envisagent d'intégrer 1 million de neurones de type *Integrate-and-fire* dans la version future de leur système. Cette grande taille des réseaux simulables se fait au détriment de la précision des modèles utilisés et la flexibilité de configuration. L'échelle temporelle des simulations est de l'ordre de $10^4$ fois le temps réel biologique (Schemmel, 2007). D'un autre côté, notre équipe réalise des circuits analogiques pour simuler des neurones selon le formalisme de *Hodgkin et Huxley* permettant la modélisation de plusieurs types de neurones complexes fonctionnant en temps réel biologique. Qualifiés de circuits biologiquement réalistes, ces modèles permettent de représenter la dynamique de la membrane cellulaire telle quelle est décrite dans la biophysique (Saïghi, 2004). Côté réseau, des modèles réalistes de synapse et de plasticité interfacent les entrées et les sorties des circuits des neurones. La connectivité et la plasticité du réseau sont entièrement configurables et permettent de spécifier des réseaux complètement connectés (*all-to-all*). La précision de calcul accompagnée de la flexibilité de configuration, qu'offre notre approche, priment sur la taille du réseau et permettent d'atteindre un degré de réalisme élevé. La taille de réseau est de l'ordre de quelques centaines de neurones.

Les deux approches sont complémentaires de point de vue de leur domaine d'application et essayent de conclure sur l'importance relative de la qualité et la quantité des cellules simulées par les systèmes neuromorphiques. Les deux approches permettent, comme le montre les premiers résultats expérimentaux (Daouzli, 2009), d'explorer des questions de neurosciences se rapportant à la dynamique des réseaux.

**Cadre scientifique**

Le travail présenté dans cette thèse s'est fait dans la continuité d'autres travaux de l'équipe. Les circuits analogiques qui représentent les neurones sont le résultat des travaux de L. Alvado (Alvado, 2003), S. Saïghi (Saïghi, 2004), T. Levi (Levi, 2007) et Y. Bornat (Bornat, 2006). Plusieurs ASICs ont été conçus et fabriqués à ce titre. Chaque nouvel circuit fabriqué augmente le potentiel du système en améliorant l'intégration et en autorisant plus de flexibilité de configuration. Le circuit *Galway* représente la dernière version d'ASIC neuromimétique et intègre 5 neurones configurables.

A partir de ces circuits, des systèmes neuromorphiques ont été conçus. Les premières versions ont été élaborées par Y. Bornat et exploitées par A. Daouzli (Daouzli, 2009). Ces systèmes étaient limités en termes de nombre de neurones simulables, ne dépassant pas la dizaine. Cette limitation provenait des grandes latences dues à la gestion logicielle de la connectivité et la plasticité du réseau. Notre contribution intervient à ce niveau, et vise à réduire les latences du système en migrant vers une réalisation purement matérielle.

**Bilan des travaux réalisés**

Au fil de ces travaux, on a augmenté le nombre de neurones simulables dans nos systèmes tout en gardant le même degré de réalisme des simulations. Deux systèmes ont été réalisés : le premier, appelé PAX2, est capable de simuler 25 neurones localisés sur une même carte. Le deuxième système, nommé PAX3, peut simuler jusqu'à 400 neurones distribués sur plusieurs cartes. Son principe consiste à distribuer le calcul sur plusieurs unités de traitement afin de pouvoir intégrer un nombre plus important de neurones.

Nos principales contributions dans la réalisation du système PAX2 sont résumées dans les points suivants :

- La conception d'une architecture numérique permettant le routage des évènements neuronaux entre les neurones localisé sur une même carte,
- La configuration du FPGA pour piloter les circuits des neurones et la communication entre la machine hôte et le système,
- L'établissement de l'environnement logiciel des systèmes (pilote de liaison série, outils de configuration et d'interprétation des données brutes),
- La conception et la réalisation du calcul de la plasticité sur FPGA.

Nos principales contributions dans la réalisation du système PAX3 sont résumées dans les points suivants :

- L'élaboration de l'architecture numérique pour assurer le dialogue entre les neurones situés sur des cartes différentes,
- La configuration des FPGAs des cartes pour piloter les circuits, gérer l'accès au bus de communication et assurer la liaison avec la machine hôte,
- L'élaboration d'un protocole de communication garantissant le retard de transit des évènements de la carte source à la carte destination en utilisant des valeurs préfixées par l'utilisateur,
- La conception et la réalisation d'un calcul de plasticité distribué sur plusieurs FPGAs.

L'évolution de la structure de gestion de réseau aboutit à un calcul réalisé en matériel, au plus près des circuits. Les latences sont par conséquent réduites et le calcul est plus rapide. Un nombre plus important de neurones peut donc être géré par le système avec le respect de l'aspect temps-réel.

Les contraintes temporelles appliquées à ces systèmes cadrent l'évolution des simulations à l'échelle temporelle des mécanismes biologiques. Ainsi, l'opération de transmission d'un évènement neuronal de la source à la destination ne doit pas dépasser une période maximale (fixée à 25 µs). Au delà de cette valeur, les résultats de simulations peuvent être altérés. De même, pour garantir la bonne utilisation des poids synaptiques de l'activation des synapses, on fixe une valeur maximale pour le traitement des poids synaptiques engendré par un potentiel d'action donné. Le principe est d'éviter l'emploi d'un poids synaptique avant sa mise à jour.

La validation du fonctionnement des systèmes fait référence à un pire cas théorique qui traduit les conditions extrêmes de l'opération des modules de calcul. L'idée derrière est de confirmer le bon fonctionnement du système s'il arrive à se comporter correctement dans le scénario du pire cas. Le pire cas se produit lorsque le réseau est complètement connecté (*all-to-all*) et les neurones génèrent simultanément leurs évènements avec des fréquences d'oscillation maximales. Le système est donc inondé par un flux d'évènements important et stressé par le nombre d'opérations de calcul qu'il doit réaliser en temps réel.

**Des simulateurs biologiquement réalistes**

Les systèmes neuronaux biologiques restent très complexes par rapport aux modèles que l'on implémente dans nos circuits. L'abstraction de certains détails et même de certains mécanismes s'avère nécessaire. Cependant, chaque abstraction doit être justifiée par l'orientation et la finalité des systèmes à concevoir.

L'équipe ISN a fait le choix de réaliser des systèmes neuromorphiques biologiquement réalistes. A cette fin, des modèles de neurones et de plasticité ont été choisis pour satisfaire le degré de réalisme souhaité. Le formalisme de Hodgkin et Huxley permet des configurations de ces paramètres biophysiques qui peuvent reproduire un grand nombre de variabilité de comportements neuronaux observables en biologie. Du côté du réseau, le modèle de plasticité tire ces origines à partir des expérimentations biologiques réalisées sur un couple de neurones isolé dans une culture *in-vitro*.

A côté des efforts des modélisateurs, le rôle des concepteurs des circuits et systèmes est important pour conserver la précision et la cinétique des modèles. Plus particulièrement, les aspects temporels d'une propagation synaptique sont étroitement liés au protocole de communication qui assure le dialogue entre les neurones. Pour simuler des réseaux de neurones biologiquement réalistes, ces phénomènes de propagation doivent être pris en compte. Les systèmes réalisés sont à la fois les révélateurs des limites et les outils d'exploration et de validation des hypothèses en neurosciences.

Des simulations biologiquement réalistes issues de réseaux composés par quelques centaines de neurones sont prévues avec le système PAX3. Ce dernier attend que les réglages des circuits des neurones soient achevés pour qu'il soit prêt à l'utilisation.

# Perspectives

### Vers une intégration d'un réseau de neurones sur ASIC

La nature prototypique des systèmes PAX2 et PAX3 fait que l'on utilise des FPGAs pour gérer les réseaux de neurones. Ce choix a été justifié par l'incertitude sur les modèles de plasticité sujets de plusieurs controverses en neurosciences. Cependant, l'utilisation des FPGA freine les performances du système en comparaison avec les circuits spécifiques, ou ASIC. En plus, les ASICs permettent des réalisations compactes et denses, ce qui rend possible l'intégration d'un réseau de neurones avec toutes ses parties constitutives dans un même circuit. L'équipe envisage une telle réalisation une fois qu'un modèle de plasticité aura l'agrément de la communauté des modélisateurs.

### Auto-calibration des modèles de neurones

Des circuits analogiques issus d'une même conception et d'un même procédé de fabrication présentent des variations au niveau des composants élémentaires. Cette spécificité s'applique aux circuits des neurones et nous amène à calibrer les neurones un par un, chacun avec un jeu de paramètres légèrement différent. La tâche est complexe si on ne procède pas à une automatisation de la calibration des paramètres. Des travaux dans cette direction ont déjà été lancés par l'équipe (Buhry, 2008) et ont permis de définir des méthodes d'auto-calibration basées sur des algorithmes heuristiques. Des optimisations sont encore nécessaires pour réduire la durée que mettent ces algorithmes pour extraire le jeu de paramètres adéquat.

### Utilisations possibles des systèmes développés

L'objectif principal de la conception de systèmes biologiquement réalistes est de fournir un outil de modélisation flexible pour les neurosciences. Les non-experts en électronique, tels que les biologistes et les informaticiens, peuvent réaliser leurs propres expérimentations par simple configuration des modules de calcul. Plusieurs applications issues de différents domaines de recherche peuvent exploiter ces systèmes :

- Un simulateur de réseaux de neurones qui opère avec un haut degré de réalisme est l'outil adéquat pour effectuer des expérimentations non réalisables en biologie. Entre autres, les phénomènes d'adaptation et d'apprentissage peuvent être étudiés avec précision. Il est également possible d'étudier l'évolution des oscillations neuronales et le codage des informations dans un réseau de neurones adaptatif.
- Un système neuromorphique fonctionnant en temps réel biologique peut être interfacé avec des réseaux de neurones vivants en boucle ouverte ou fermée (Bontorin, 2007). Le système peut jouer le rôle à la fois d'un stimulateur des synapses vivantes et d'un dispositif d'acquisition des réponses des cellules en culture, voir *in-vivo*. La partie artificielle étant maîtrisée, des analyses post-simulation permettent de tirer des conclusions à propos de mécanismes neuronaux mis en œuvre.
- Un réseau de neurones de taille suffisante doit permettre le contrôle d'un système sensorimoteur (robot). Une telle plateforme permet d'observer le réseau pendant une phase d'apprentissage en associant des stimuli consécutifs à l'évolution du système. La différentiation entre punition et récompense, ou encore l'évolution

d’un réseau en fonction de l’environnement extérieur sont autant de fonctions adaptatives intéressantes dans des systèmes automatisés.

La liste des applications réalisables avec les systèmes développés n’est pas exhaustive. Les moyens de notre l’équipe ne lui permettent pas de tout suivre. A court et moyen termes, l’équipe a choisi de mettre l’accent sur les deux premières applications, ce qui crée un cadre pluridisciplinaire qui fait appel à de nombreuses compétences, au-delà de l’électronique. Une plateforme de travail qui réunit les chercheurs de différentes disciplines s’avère nécessaire. Cette plateforme s’est récemment concrétisée par la fusion de l’équipe *Ingénierie des Systèmes Neuromorphiques* (composée d’électroniciens, informaticiens et mathématiciens) et l’équipe *Bio-EM* (composée par des physiciens et des biologistes), ainsi que par la mise en place de projets de recherche nationaux et internationaux pluridisciplinaires.

# Bibliographie

Abbott L. F., Single neuron dynamics : an introduction, in neural modelling in neural network, *Pergamon Press, Oxford*, 1994.

Abeles M, Local cortical circuits : An electrophysiological study. *Series of brain function*, v. 6. *Springer-Verlag*, ISBN 0387110348, 1982.

Alibart F., Pleutin S., Guérin D., Novembre C., Lenfant S., Lmimouni K., Gamrat C., Vuillaume D., An organic nanoparticule transistor behaving as a biological spiking synapse, *Advanced Functional Materials journal*, 20: 330–337, 2010.

Alvado L., Neurones artificiels sur silicium: une évolution vers les réseaux. *Manuscrit de thèse, Université Bordeaux 1*, 2003.

Amari S., Arbib M. A., Competition and cooperation in neural nets, *J. Metzler editor. Systems neuroscience. Academic press*, 65–119, 1977.

Belhadj B., Tomas J., Un système mixte de traitement de l'information neuronale destiné à la simulation des réseaux de neurones réalistes, *JNRDM'08*, 2008a.

Belhadj B., Tomas J., Mallot O., N'Kaoua G., Bornat Y. and Renaud S., FPGA-based architecture for real-time synaptic plasticity computation, *ICECS*, 93-96, 2008b.

Belhadj B., Tomas J., Mallot O., N'Kaoua G., Bornat Y. and Renaud S., Token-passing communication protocol in hardware based real-time spiking neural network, SPIE, 1-11, 2009a.

Belhadj B., Tomas J., Mallot O., Daouzli A., Bornat Y. and Renaud S., Digital mapping of a realistic spike timing plasticity model for real-time neural simulation, DCIS, 326-331, 2009b.

Belhadj B., Tomas J., Mallot O., Bornat Y., N'Kaoua G. and Renaud S., Guaranteeing spike arrival time in multiboard & multichip spiking neural networks, ISCAS, 377-380, 2010.

Bell C. C., Han V. Z. and Sugawara, Synaptic plasticity in a cerebellum-like structure depends on temporal order. *Nature*, 387: 278–281, 1997.

Bi G. and Poo M., Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *Neural Computation*, 9: 503–514, 1997.

Bi G. and Poo. Synaptic modification in cultured hippocampal neurons: dependence on spike timing, synaptic strength and postsynaptic cell type, *Journal of neuroscience*, 18(10): 464–472, 1998.

Binzeger T., Douglas R. J., Martin K., A quantitative map of the circuit of cat primary visual cortex. *Journal in Neuroscience*, 24(39): 53–8441, 2004.

Bliss T. V. P. and Gardner-Medwin A. R., Long-lasting potentiation of synaptic transmission in the dendate area anasthetized rabbit following stimulation of the perforant path, *Journal of physiology*, 232:357–374, 1973.

Boahen K., Communicating neural ensembles between neuromorphic chips, *Neuromorphic systems engineering*, 229–259, 1998.

Boahen K., http://www.stanford.edu/group/brainsinsilicon/neurogrid.html, 2006.

Bontorin G., Renaud S., Garenne A., Alvado L., Le Masson G., and Tomas J., A real-time closed-loop setup for hybrid neural networks. *Proceedings of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS2007)*, 2007.

Bornat Y., Réseaux de neurons sur silicium : une approche mixte, analogique / numérique, pour l'étude des phénomènes d'adaptation, d'apprentissage et de plasticité. *Manuscrit de thèse*, *Université de Bordeaux*, 2006.

Bornat Y., Renaud S., Tomas J., Saighi S., Zou Q., Destexhe A.. An analog/digital simulation system for biomimetic neural networks, *EPFL Latsis Symposium 2006, Dynamic principles for neuroscience and intelligent biomimetic devices*, Lausanne, 2006bis.

Bornat Y., Tomas J., Lopez C., Malot O., Belhadj B. and Renaud S., Design of analog/digital simulator dedicated to real-time neurocomputing. *In DCIS'07*, 391-396, 2007.

Briggman K. L. and Denk W., Towards neural circuit reconstruction with volume electron microscopy techniques. *Current Opinion in Neurobiology*, 16(5):562–570, 2006.

Bury L., Saighi S., Giremus A., Grivel E., and Renaud S., Parameter estimation of the Hodgkin-Huxley model using metaheuristics: application to neuromimetic analog integrated circuits. *IEEE Biomedical circuits and systems conference*, 173–176, 2008.

Calvin W. H. and Stevens C. F., Synaptic noise as a source of variability in the interval of action potentials. *Science*, 155:842-844, 1967.

Calvin W. H. and Stevens C. F., Synaptic noise and other sources of randomness in motoneuron interspike intervals. *Science*, 31:574-587, 1968.

Carlo J. T., Understanding token ring protocols and standards, Artech House editor, ISBN 089006458X, 1998.

Cauwenberghs G. and Bayoumi M. A., *Learning on Silicon: Adaptive VLSI Neural Systems*, pages 1–29, Norwell, MA, 1999.

Cohen D., Magnetoencephalography: Evidence of magnetic fields produced by alpha-rhythm currents. *Science*, 161(3843):784–786, 1968.

Chicca E., Whatley A. M., Dante V., Lichtsteiner P., Delbürck T., Del Giudice P., et. all, A multi-chip pulse-based neuromorphic infrastructure and its application to a model of orientation selectivity, *IEEE circuits and systems*, 5(54): 81–93, 2007.

CNE, Cognitive Neuromorphic Engineering, *Sardigna Workshop*, 2009.

Culurciello, E. and Andreou, A. G., A comparative study of access topologies for chip-level address-event communication channels, *IEEE Transactions on Neural Networks*, 1266-1277, 2003.

Dally W. J.and Poulton J. W., *Digital systems engineering*. Cambridge University Press, New York, NY, USA, 1998. ISBN 0-521-59292-5.

Daouzli M. A., Systèmes neuromorphiques : étude et implantation de fonctions d'apprentissage et de plasticité, rapport de thèse, *université de Bordeaux 1*, 2009.

Dayan P., Abbott L., Theoretical neuroscience: computational and mathematical modelling of neural systems, *Cambridge MA: MIT press*, 2001.

Delbrück T. and Liu S. C., A silicon early visual system as a model animal. *Vision Res*, 44(17):2083-2089, 2004.

Destexhe A., Conductance-based integrate-and-fire models. *Neural Comput.*, 9(3):503–514, 1997. ISSN 0899-7667.

Destexhe A., Mainen Z. F., Sejnowski T. J., Kinetic models of synaptic transmission, *in Methods in Neural Modeling*, Eds. C. Koch et I. Segev, MIT Press, Cambridge MA, 1999.

Destexhe A. and Mainen Z. F., Palsticity in single neuron and circuit computation, *Nature*, 6: 789–795, 2004.

Douglas R., Mahowald M., Mead C., Neural circuits of the neocortex. *Annl Rev Neurosci.*, 27: 51–419, 2004.

Farquhar I., Gordon C., Hasler P., A field programmable neural array, *ISCAS'06*, 4117–4120, 2006.

Froemke R. C. and Dan Y., Spike-timing-dependent synaptic modification induced by natural spike trains, *Nature*, 416: 433–438, 2002.

Froemke R. C., Poo M. and Dan Y.. Spike-timing-dependent plasticity depends on dendritic location, *Nature*, 434: 221–225, 2005.

Gerstner W. and Kistler W., *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

Häfliger P., Adaptive WTA with analog VLSI neuromorphic learning chip. *IEEE transactions on neural networks*, 18(2):51-72, 2007.

Hebb D. O., The organization of behaviour: a neurophysiological theory. *ISBN 1-4106-1240-6*, 1949.

Hodgkin A. L. and Huxley A. F., A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of physiologyI*, 116:500–544, 1952.

Izhikevich E. M., Which model to use for cortical spiking neurons ?, *IEEE transactions in neural networks*, 15(5): 1063–1070, 2004.

Johansson C. and Lansner A.. Towards cortex sized artificial neural systems. *Neural Networks*, 20(1):48–61, 2007.

Jolivet R.,. Kobayashi R, Rauch A., Naud R., Shinomoto S., and Gerstner W., A benchmark test for a quantitative assessment of simple neuron models. *Journal of Neuroscience Methods*, 169(2):417 – 424, 2008. ISSN 0165-0270. Methods for Computational Neuroscience.

Jung R., Berger W., and Berger H., Fiftieth anniversary of Hans Berger's publication of the electroencephalogram. *Arch Psychiatr Nervenkr*, 227:279–300, 1979.

Lampl I., Reichova I., and Ferster D., Synchronous membrane potential fluctuations in neurons of the cat visual cortex. *Neuron*, 22(2):361–74, 1999.

Levi T., Méthodologie de développement d'une bibliothèque d'IP-AMS en vue de la conception automatisée de systèmes sur puces analogiques et mixtes: application à l'ingénierie neuromorphique, *Manuscrit de thèse*, 2007.

Lewis M. A., Etienne-Cummings R., Cohen A. H. and Hartmann M., Toward biomorphic control using custom a VLSI chips. *Proceedings of the International conference on robotics and automation*, IEEE press, 2000.

Lichtsteiner P. and Delbruck T., 64x64 event-driven logarithmic temporal derivative silicon retina. *IEEE workshop on charge-coupled devices and advanced image sensors*, 157-160, 2005.

Lin J., Merolla P., Arthur J., Boahen K., Programmable connections in neuromorphic grid. *49th Midwest Symposium on Circuits and Systems*, 2006.

Logothetis N. K. Pauls J., Augath M., Trinath T. and Oeltermann A., Neurophysiological investigation of the basis of the fMRI signal. *Nature*, 412(6843):150–157, 2001.

Lountcastle V., The columnar organization of the neocortex. *Brain*, 120, 701–722, 1997.

Magee J. C. and Johnston D., A synaptically controlled associative signal for Hebbian plasticity in Hippocampal neurons. *Science*, 275: 209–213, 1997.

Mahowald M, VLSI analogs of neuronal visual processing: A synthesis of form and function. Rapport de thèse, *California Institute of Technology*, Pasadena,1992.

Malcolm N. and Zhao W., The timed-token protocol for real-time communications, *IEEE Computer*, 35-41, 1994.

Malcolm N. and Zhao W., Guaranteeing synchronous messages with arbitrary deadline constraints in an FDDI network, IEEE Computer, 186-195, 1993.

Markram H., Wang Y., and Tsodyks M., Differential signaling via the same axon of neocortical pyramidal neurons. *Proceedings of the National Academy of Sciences of the United States of America*, 95(9):5323–5328, ISSN 0027-8424, 1998.

Mead C. A. and Mahowald M. A., A silicon model of early visual processing. *Neural Networks*, 1(1):91–97, 1988.

Mead C. A., Neuromorphic electronic systems. *Proceedings IEEE*, 78:1629–1636, 1990.

Merolla P. A. and Boahen K., Dynamic computation in a recurrent network of heterogeneous silicon neurons. *Proceedings of the 2006 IEEE International Symposium on Circuits and Systems (ISCAS '06)*, 2006.

Morrison A., Mehring C., Geisel T., Aertsen A., and Diesmann M., Advancing the boundaries of high connectivity network simulation with distributed computing. *Neural Comput.*, 17 (8):1776–1801, 2005.

Nam Y., Brown E. A., Ross J. D., Blum R., Wheeler B. C. and DeWeerth S. P., A retrofitted neural recording system with a novel stimulation IC to monitor early neural responses from a stimulating electrode. *Journal of Neuroscience Methods*, 178:99:102, 2009.

Pleinevaux, P., Real-time fault tolerant operation of the 802.5 token ring, Real-time systems, 79-91, 2005.

Purschke M., Kandasamy A., Kriplani A., Lecomte R., O'Connor P., Pratte J.-F., Radeka V., Schlyer D., Southekal S., Stoll S., Vaska P., Villanueva A., Woody C., Junnakar S., Krishnamoorthy S., Shokouhi S., Fontaine R., and Dzhordzhadze V., The ratcap conscious small animal pet tomography. *IEEE-NPSS Real Time Conference*, 2005.

PyNN, langage de spécification des réseaux de neurones : http://neuralensemble.org/trac/PyNN/, 2005.

Ramon y Cajal S., *Histologie du Système Nerveux de l'homme et des Vertébrés*, volume 2. 1911.

Renaud S., Tomas J., Bornat Y., Daouzli A., and Saïghi S., Neuromimetic ICs with analog cores: an alternative for simulating spiking neural networks. *Proceedings of the 2007 IEEE Symposium on Circuits and Systems (ISCAS'07)*, 2007.

Renaud S., Thomas J., Lewis N., Bornat Y., Daouzli A., Rudolph M., Destexhe A. and Saïghi S., PAX: a mixed hardware/software simulation platform for spiking neural networks, *Neural Networks*, accépté pour publication journal, 2010.

Sakmann B. and Neher E., Single-channel recording, *Plenum press*, 1995.

Saïghi S., Circuits et systèmes de modélisation analogique de réseaux de neurones biologiques : application au développement d'outils pour les neurosciences computationnelles, *Manuscrit de thèse, Université Bordeaux 1*, 2004.

Saïghi S., Bornat Y., Tomas J. and Renaud S., Neuromimetic ICs and system for parameters extraction in biological neural models, *ISCAS'06*, 4207-4210, 2006.

Schemmel J., Brüderle D., Meier K., and Ostendorf B., Modeling synaptic plasticity within networks of highly accelerated I&F neurons. *Proceedings of the 2007 IEEE International Symposium on Circuits and Systems (ISCAS'07)*. IEEE Press, 2007.

Schemmel J., Fieres J., and Meier K., Wafer-scale integration of analog neural networks. In *Proceedings of the 2008 International Joint Conference on Neural Networks (IJCNN)*, 2008.

Schwartz M., Telecommunication networks: Protocols, Modeling, and Analysis, *Addison-Wesley, Reading*, MA, 1987.

Sevicik K. C. and Johnson M. J., Cycle time properties of the FDDI token ring protocol, *IEEE Transactions on Software Engineering*, 376-385,1987.

Shelley M., McLaughlin D., Shapley R., and Wielaard J., States of high conductance in a large-scale model of the visual cortex. *J. Comp. Neurosci.*, 13:93–109, 2002.

Sjöström J., et al., Rate, timing, and cooperativity jointly determine cortical synaptic plasticity. *Neuron 32*, 1149—1164, 2001.

Tanenbaum, A. S., Computer Networks, Prentice-Hall International, *Upper Saddle River*, NJ, 2 edition, 1989.

Tritsch D., Chesnoy-Marchais D. and Feltz A., Physiologie du neurone. *Unitiatives santé*, ISBN 2-7040-0872-8, 1998.

Turing A., On computable numbers, with an application to the Entscheidungs problem. *Proceedings of the London Mathematical Society*, 42:230–265, 1937.

Vogelstein R. J., Mallik U., Vogelstein J. T., Cauwenberghs G., Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses, *IEEE transactions on neural networks*, 1(18): 253–265, 2007.

Wendy M. H., William J. C., Karl, F. P., FDDI: an introduction to fiber distributed data interface. ISBN: 15555800939, 1993.

Xilinx, DS099-1 (v1.1): Spartan-3 1.2V FPGA Family: Introduction and ordering information. Advance product specification, April 24, 2003.

Yao L. J., Zhao W., Lim C. C., A comparative study of three token ring protocols for real-time communications, *International journal of mini & microcomputers*, vol.17, n°2, pp 84-97, 1995.

# Publications de l'auteur

## Conférences internationales avec comité de lecture

Y. Bornat, J. Tomas, C. Lopez, O. Malot, B. Belhadj and S. Renaud, ***Design of analog/digital simulator dedicated to real-time neurocomputing***, DCIS, 391-396, Seville, 2007.

B. Belhadj, J. Tomas, O. Malot, G. N'Kaoua Y. Bornat, and S. Renaud, ***FPGA-based architecture for real-time synaptic plasticity computation***, ICECS, Malta, 2008.

B. Belhadj, J. Tomas, O. Malot, Y. Bornat, S. Saïghi and S. Renaud, ***Plasticity computation in a multiboard system simulator dedicated to SNN,*** Neurocomp, Bordeaux, 2009.

B. Belhadj, J. Tomas, O. Malot, G. N'Kaoua Y. Bornat, and S. Renaud, ***Token-passing communication protocol in hardware based real-time spiking neural network***, SPIE, Dresden, 2009.

B. Belhadj, J. Tomas, O. Malot, A. Daouzli, Y. Bornat, and S. Renaud, ***Digital mapping of a realistic spike timing plasticity model for real-time neural simulation***, DCIS, Zaragoza, 2009.

B. Belhadj, J. Tomas, O. Malot, Y. Bornat, G. N'Kaoua, and S. Renaud, ***Guaranteeing spike arrival time in multiboard & multichip spiking neural networks***, ISCAS, Paris, 2010.

S. Saïghi, J. Tomas, Y. Bornat, B. Belhadj, O. Malot and S. Renaud, ***Real-time multiboard architecture for analog spiking neural network***, Special session, ISCAS, Paris, 2010.

S. Saïghi, T. Levi, B. Belhadj, O. Malot and J. Tomas, ***Hardware system for biologically realistic, plastic, and real-time spiking neural network simulations***, accepted in IJCNN, Barcelona, 2010.

## Conférence nationales avec comité de lecture

B. Belhadj et J. Tomas, ***Un système mixte de traitement de l'information neuronale destiné à la simulation des réseaux de neurones réalistes***, JNRDM, Bordeaux, 2008.

## Rapports techniques

J. Tomas, O. Malot, B. Belhadj and S. Renaud, ***Technical report on the second generation platform***, D6_1, FACETS EU project, September 2007.

J. Tomas, O. Malot, B. Belhadj and S. Renaud, ***Technical report on the multiboard simulation platform***, D6_3, FACETS EU project, September 2008.

B. Belhadj, J. Tomas, O. Malot, and S. Renaud, ***Plasticity function implementation***, Technical report, Neurovers-IT EU project, June 2009.

B. Belhadj, J. Tomas, O. Malot, and S. Renaud, ***Second Technical report on the multiboard platform***, D6_6, FACETS EU project, July 2010.

# Annexe A : Configuration et communication dans le système PAX2

Des outils logiciels ont été développés pour piloter le système PAX2 à partir d'une machine hôte. Ces outils reposent sur des protocoles de configuration et de communication que l'on a définie. Cette annexe détaille le mode de fonctionnement de ces protocoles.

## A.1 Supports de communication

Le système PAX2 est relié à un ordinateur de commande par l'intermédiaire de trois ports série de type RS232 dont la répartition de leur utilisation est comme suit :

- Le port *RS_0* : est utilisé pour assurer la configuration des paramètres des ASICs (paramètres analogiques et numériques). Il retourne des informations sur l'état du module de la gestion du bruit synaptique (voir plus loin). Le débit de fonctionnement est de 38400 bits/s.
- Le port *RS_1* : assure l'envoi des signaux de contrôle au système, tels que les signaux *reset* et *start_sim*. Il permet également de configurer la connectivité et la plasticité du réseau. Dans l'autre sens, il permet de récupérer une partie des résultats de simulation. Il fonctionne avec un débit de 921600 bits/s.
- Le port *RS_2* : assure la configuration du module du bruit synaptique au cours de la simulation. Dans l'autre sens, il permet de récupérer une partie des résultats de simulation. Il fonctionne avec un débit de 921600 bits/s.

Les étapes de la gestion de la communication du système avec la machine hôte sont résumées dans la figure A.1.

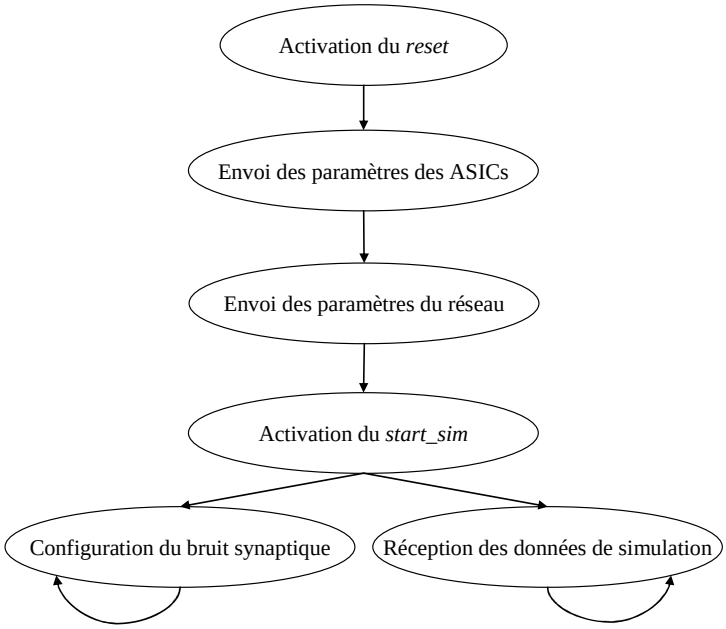



Figure A.1. Etapes de la communication entre le système PAX2 et la machine hôte.

## A.2 Configuration du système

La configuration dans le système PAX2 concerne 3 modules de calcul : les ASICs (les circuits *Galway*), le module de gestion du réseau et le module du bruit synaptique. La sélection du module se fait par l'insertion d'un code spécifique codé sur 2 bits. La taille de la trame de donnée est relative au module du calcul.

*Configuration des neurones*

La configuration des neurones consiste à envoyer une série de paramètres aux ASICs. Les paramètres peuvent être numériques ou analogiques. La trame de configuration des neurones devrait spécifier l'adresse de l'ASIC (entre 1 et 5), le numéro du neurone dans l'ASIC (entre O et 4), le type de paramètre (numérique ou analogique) et sa valeur. La figure A.2 résume le format nécessaire pour une trame de configuration.

| Code (2bits) | No ASIC (3bits) | No neurone (3bits) | Type donnée (1 bit) | Donnée (14 bits) |
|---|---|---|---|---|

Figure A.2 Trame de configuration des neurones.

Les modules qui reçoivent cette trame ont un accès direct aux ASICs. Ils interprètent les champs de la trame agissant selon leur contenu. Dans le cas où le paramètre est numérique, une séquence de 14 bits sera envoyée au neurone. Un exemple d'envoi d'un paramètre numérique est fourni dans la figure A.3. Une horloge accompagne chaque donnée sur 14 bits pour synchroniser la réception. Un signal de validation indique la fin de la transmission du paramètre. Ces paramètres vont par la suite agir sur l'état des interrupteurs au sein des circuits permettant de définir la topologie des neurones de point de vue canaux ioniques.

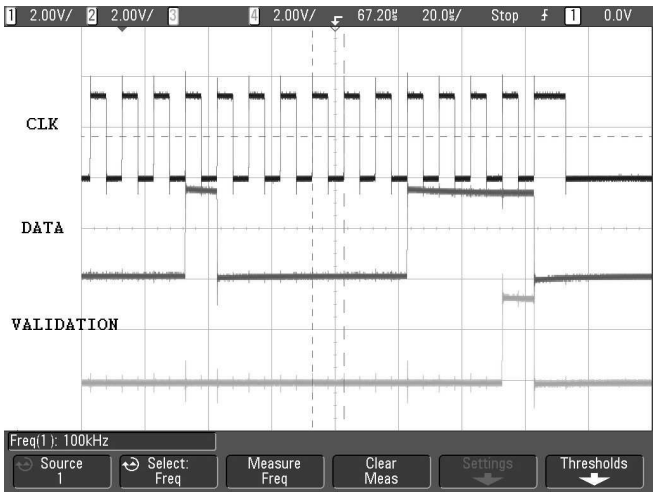



Figure A.3. Illustration de la configuration des circuits *Galway* par l'intermédiaire des paramètres numériques.

Une fois tous les paramètres numériques sont envoyés, l'envoi des paramètres analogiques débute. Le module qui gère cet envoi redirige la valeur de la donnée vers les DACs situés juste avant l'entrée des ASICs. Les valeurs des paramètres sont donc transformées en leur équivalent en analogique à l'entrée des circuits. Des signaux de contrôle suivent cette

opération pour synchroniser la réception. La figure A.4 fournit un exemple illustratif de cette opération.

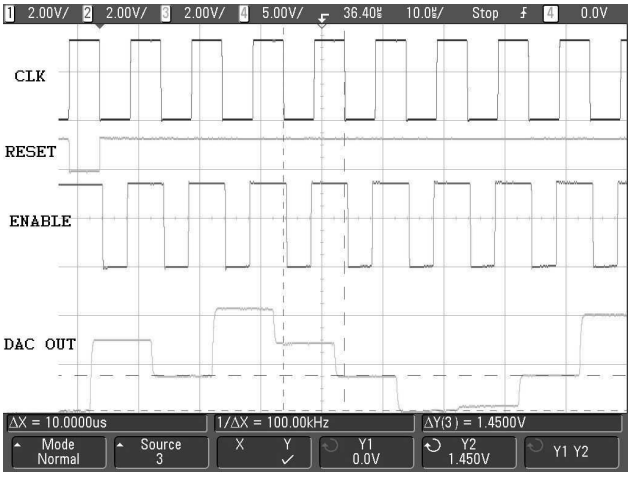



Figure A.4. Chronogramme des entrées des paramètres analogiques au niveau du circuit *Galway*.

Pour un seul circuit, on envoi 205 paramètres analogiques pour une configuration complète. Les paramètres sont stockés dans des mémoires analogiques dans le circuit. Ces mémoires doivent être rafraîchies périodiquement pour garder les bonnes valeurs des potentiels. Le rafraîchissement est donc réalisé par le renvoi des mêmes séquences de paramètres chaque période de 2 ms.

*Configuration du réseau*

La configuration du réseau suppose que les matrices de configuration sont déjà remplies par l'utilisateur final. Ces matrices seront interprétées par un programme informatique qui les traduit en trames destinées au module de gestion de réseau. Les données correspondant à la connectivité ont la forme suivante :

@ du neurone : |*liste des neurones postsynaptiques*| FF |*liste des neurones présynaptiques*|

Pour chaque neurone, on définit les listes de ses neurones présynaptiques et postsynaptiques séparées par un mot réservé (FF). Ces paramètres sont sauvegardés dans une mémoire RAM appelée RAM de connectivité. Elle contient 2024 entrées de 8 bits chacune. Les données relatives à un neurone sont stockées sur 64 entrées. La figure A.5 illustre un exemple de configuration d'un neurone qui est connecté à 3 neurones présynaptiques et 2 postsynaptiques (valeurs en hexadécimale).

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 01 | 12 | 09 | FF | 01 | 05 | FF | … | FF |

Figure A.5. Configuration de la connectivité d'un neurone : la liste de neurones présynaptiques (les neurones 01, 12 et 09) et séparée de la liste des neurones postsynaptiques (les neurones 01 et 05) par un FF. Les cases restantes des 64 entrées réservées à la configuration du neurone sont toutes mises à FF.

En ce qui concerne la configuration de la plasticité de réseau, une deuxième mémoire RAM, appelée RAM des poids, sauvegarde les poids synaptiques et la plasticité des connexions. Elle est organisée de la même façon que la RAM de connectivité à l'exception

que les entrées sont codées sur 17 bits. Les 16 premiers bits accueillent les poids synaptiques, alors que le 17ème bit configure la plasticité de la connexion.

Un registre de 32 bits permet de configurer la polarité des neurones. Le numéro de bit correspond au numéro de neurone.

Pour distinguer les différents paramètres de configuration au niveau du FPGA, un caractère distinctif est ajouté en amont de chaque octet de donnée envoyé. Ainsi, le fichier de configuration ressemble à l'illustration de la figure A.6 (valeurs en hexadécimale).

```
c01c12c09cFFc01c05cFFcFFcFF…w80w01w54w01w78w00wFFwFFw47w01w74w00wFF…s01s00s18s00.
```

Figure A.6. Exemple du contenu d'un fichier de configuration : distinction entre les paramètres des RAMs. Les données destinées à la RAM de connectivité sont précédées par le caractère 'c', celles pour destinées à la RAM des poids commencent par le caractère 'w'. Le caractère 's' sert pour distinguer les données de polarité.

*Configuration du module du bruit synaptique*

A la différence des autres configurations, la configuration du module du bruit synaptique continue à fonctionner durant la simulation. Les données de configuration seront accueillies par plusieurs mémoires RAMs. L'organisation d'une RAM est illustrée par la figure A.7. Un bloc RAM est divisé en deux parties, chacune de 512 entrées de 16 bits destinés à recevoir les ISIs. Chaque partie contient les données pour un neurone et est divisée en deux compartiments : un compartiment pour stocker les valeurs des ISIs de la synapse excitatrices, et un deuxième pour la synapse inhibitrice.

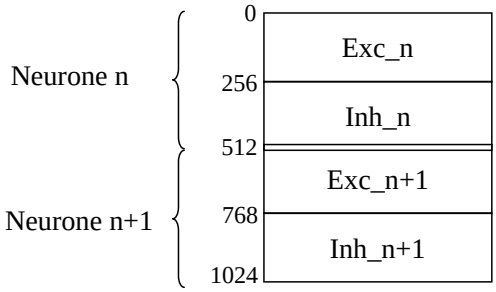



Figure A.7. Organisation d'une RAM de stockage des ISIs.

Les valeurs de chaque compartiment sont rafraîchies périodiquement. Un protocole est défini pour assurer cette tâche. La figure A.8 décrit son fonctionnement. Le FPGA réalise une demande de rafraîchissement en envoyant l'identificateur de la RAM (*RAM_id*), le numéro du compartiment (*cprt*) et l'adresse à partir de laquelle il commence le rafraîchissement. La machine hôte répond par l'envoi de 256 nouvelles valeurs à insérer dans le compartiment en question, et finie par une l'envoi d'un signal de fin de l'opération. Cette opération se répète jusqu'à la fin de la simulation. Le rafraîchissement des RAMs se fait en tour de rôle.

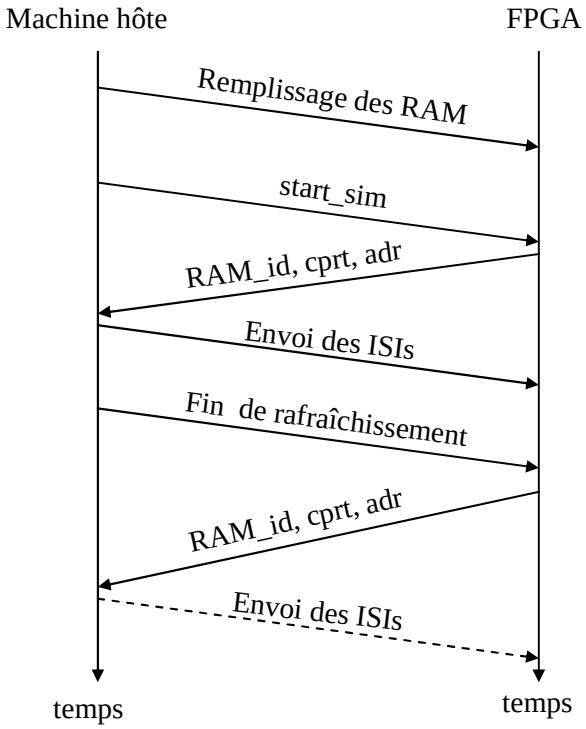



Figure A.8. Protocole de la configuration périodique des données du module du bruit synaptique.

Pour un rafraîchissement, l'ordinateur envoie une trame selon le format suivant :

| RAM_id | cprt | Adr_1 | Donnée_1 | Adr_2 | Donnée_2 | … | Adr_256 | Donnée_256 |
|---|---|---|---|---|---|---|---|---|

## A.3 Récupération des résultats de simulation

Les résultats de simulation que l'on souhaite récupérés sont les valeurs de poids synaptiques et l'activité des neurones. Il y a donc deux types de trame à récupérer, dont les formats sont donnés si dessous :

Trame de l'activité des neurones :

| Adr_nrn (8 bits) | Heure d'apparition (31 bits) | Bit de trame (1 bit) |
|---|---|---|

Trame des poids (ou connexions) :

| Adr_pre (8 bits) | Adr_post (8 bits) | Poids (8 bits) | Heure (15 bits) | Bit de trame (1 bit) |
|---|---|---|---|---|

Ces trames seront envoyées par le FPGA par les liaisons *RS_1* et *RS_2*. Pour maximiser le débit de transmission, on a alterné l'envoie des données sur les deux ports. L'ordinateur reçoit les deux types de trames chevauchées. Pour pouvoir les différencier, on affecte différemment le dernier bit de chaque trame (*Bit de trame*).

# Annexe B : Configuration et communication dans le système PAX3

Le système PAX3 est composé de plusieurs cartes connectées entre elles par l'intermédiaire d'un fond de panier. Une carte spéciale, appelée carte mère, agit comme une interface entre le système et la machine hôte.

## B.1 Support de communication

Une liaison USB permet la communication entre le système et l'ordinateur. A la différence de la liaison série RS232, la liaison USB ne permet que l'envoi et la réception dans un seul sens à la fois. Ceci ne permet plus de réaliser des configurations au cours des simulations puisque le port USB sera occupé par la récupération des données de réception. Les phases de communication sont donc constituées d'une phase de configuration générale suivie d'une phase de simulation. La figure B.1 résume les étapes suivies dans ces phases. Le débit de la liaison USB est 25 Mo/s.

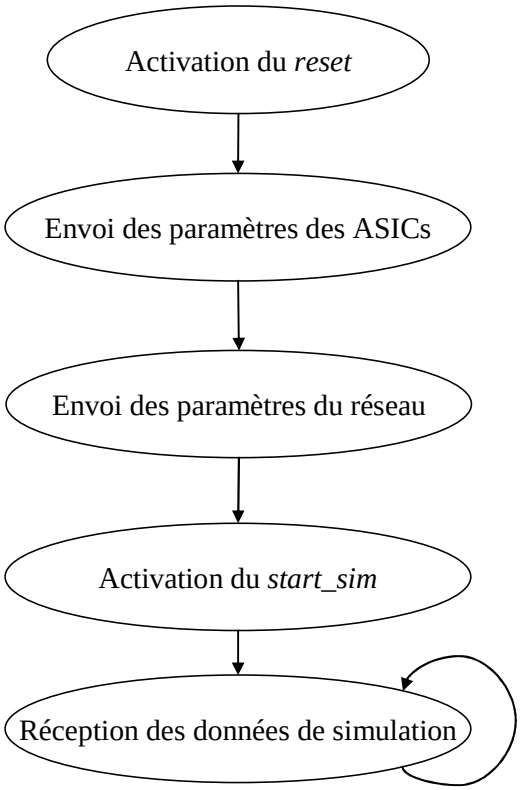



Figure B.1. Etapes de communication entre le système PAX3 et la machine hôte.

La carte mère distribue les données de configuration sur les autres cartes du système en utilisant un bus de communication commun. Le bus de communication peut véhiculer jusqu'à 68 bits en parallèle avec un débit de 25 Mbits/s par fil, soit un débit maximal de 1,7 Gbits/s. Des fils spéciaux, au nombre de cinq, servent à la réalisation des connexions point-à-point enter les cartes. Ils sont appelés les fils de *daisy-chain*. En dehors de la phase de configuration, ce bus est utilisé comme support pour la communication inter-neurones.

Les fils du bus de communication ne sont pas tous utilisés pour envoyer les données. Ils sont répartis sur plusieurs fonctions de communication. Le tableau B.1 fournie la fonctionnalité de chaque des fils du bus.

Tableau B.1. Répartition des ressources du bus de communication.

| **Ressources** | **Fonction** |
|---|---|
| 1 fil parallèle | Mise à zéro du système. |
| 1 fil parallèle | Délimite les phases de simulation et configuration |
| 1 fil parallèle | Indique la présence d'une donnée |
| 1 fil parallèle | Indique le type de messages (MTR ou MEM) |
| 1 fil parallèle | Horloge de synchronisation |
| 51 fils parallèles | Envoi des trames de donnée et de configuration |
| 3 daisy-chain | Programmation des FPGA par le port JTAG |
| 1 daisy-chain | Anneau à jeton |
| 10 fils parallèles | Fils prévus pour une extension du système |

## B.2 Configuration du système

Les trames de configuration prennent en compte la structure multicarte du système. Leur entête commence par le numéro de la carte. Ensuite, leur destination à l'intérieur de la carte est déterminée par un code spécifique. Le format d'une trame de configuration est comme suit :

| Numéro de carte (5 bits) | Code composant (3 bits) | données (43 bits) |
|---|---|---|

Le nombre total de bits est de 51 (la largeur de bus de donnée). Le code composant permet de sélectionner le traitement des données dans le FPGA. Il existe 6 composants configurables dans le système PAX3 :

- Composant de numérotation : il permet d'affecter des numéros aux neurones d'une carte. Cette opération s'effectue par le remplissage de la table de conversion. Les données sont une séquence de 3 numéros codés sur 9 bits chacun. Le remplissage de la table de conversion nécessite 10 trames.

- Composant du processeur : le processeur du calcul de la plasticité est configuré par le nombre de neurones qui existe dans le réseau et les paramètres $w_{LTP}$ et $w_{LTD}$. Il nécessite 2 trames de configuration.
- Composant du calcul permanant : il s'agit du composant qui calcule les exponentielles. Il y a 4 exponentielles à configurer. Pour chaque exponentielle, il faut configurer les paramètres $P_0$ et l'amplitude, ce qui conduit à l'envoi de 8 trames de configuration.
- La RAM de plasticité : il s'agit de la configuration des poids synaptiques initiaux, de la plasticité des connexions et de la polarité des synapses. Une trame permet la configuration de 2 entrées de la RAM. Ce qui revient à envoyer 5120 trames de configuration pour finir la configuration du composant. Cependant, il n'est pas nécessaire de configurer les entrées qui ne seront pas utilisées durant la simulation (absence de connexion) puisqu'elles sont mises à zéro par défaut.
- Le composant de la communication : le module qui gère le protocole de communication reçoit une trame de configuration pour initialiser les paramètres TTRT et $THT_{max}$.
- Les ASICs : les circuits des neurones sont configurés par les paramètres analogiques et numériques. Ces paramètres sont utilisés de la même façon que dans le système PAX2. La composition de la séquence de donnée ne change pas non plus. Une trame permet de configurer un paramètre. Comme on a besoin de 60 (3mots par neurone) paramètres numériques et de 820 (205 paramètres par ASIC), on aura besoin au total de 880 trames de configuration.

## B.3 Récupération des données

Les données récupérées sont les activités des neurones et les poids synaptiques. Il est donc nécessaire d'écrire ces données sur le bus de communication commun pour qu'elles soient écoutées par la carte mère. L'activité des neurones (l'adresse des potentiels d'action + l'heure d'apparition) est naturellement transmise sur le bus pour assurer l'interaction entre les neurones pendant la phase de simulation. Ces données sont donc récupérées à la volée par la carte mère est envoyées à la machine hôte. Les poids synaptiques sont calculés et utilisés localement dans chaque carte. Leur transmission sur le bus de communication est nécessaire afin d'être envoyés vers la machine hôte. Le protocole de communication entre les cartes permet d'allouer de la bande passante pour la transmission de ces messages. L'envoi est effectué selon la politique *best-effort.*

Les trames des données sont sur 52 bits. La distinction entre les deux types de trames se fait par l'activation d'un bit supplémentaire qui passe à '1' logique lorsqu'il s'agit d'envoyer les données de l'activité. Les formats des trames sont fournis ci-dessous :

Trame des potentiels d'action (ou message MTR) :

| Nrn1 (9 bits) | Heure (8 bits) | Nrn2 (9 bits) | Heure (8 bits) | Nrn3 (9 bits) | Heure (8 bits) |
|---|---|---|---|---|---|

Trame des poids (ou message MEM) :

| Adr_pre (8 bits) | Adr_post (8 bits) | Poids (8 bits) | Heure (27 bits) |
|---|---|---|---|

La trame des potentiels d'action peut contenir de 1 à 3 évènements. Dans le cas où la trame ne contient qu'un seul évènement, les autres champs doivent être mis à zéro. L'heure est codée sur 8 bits et permet de compter jusqu'à 256 µs. La carte mère concatène les 24 bits de poids forts de l'heure absolue aux 8 bits de la trame pour reconstituer l'heure de l'apparition de l'évènement. Les trames envoyées à la machine hôte seront, par conséquent, composées de 40 bits.