Modèles cellulaires de champs neuronaux dynamiques

Benoît Chappet de Vangel

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Modèles cellulaires de champs neuronaux dynamiques

THÈSE

présentée et soutenue publiquement le 15 Septembre 2016

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Benoît Chappet de Vangel

Composition du jury

Président :	Le président
Rapporteurs :	Pr. Benoît Miramond Pr. Jean-Pierre Dérutin
Examinateurs :	Dr. Yulia Sandamirskaya Dr. Joseph Luis Rosselló Pr. Emmanuel Jeandel
Encadrant de thèse :	Pr. Bernard Girau

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Remerciements

Je remercie tout d'abord Bernard, pour sa motivation, son soutien et pour m'avoir toujours fait confiance.

Il m'a laissé libre d'explorer les pistes que je voulais en critiquant toujours de façon constructive quand il le fallait.

Je remercie mes rapporteurs François et Benoît pour avoir accepté de lire mon manuscrit et pour avoir apporté des commentaires constructifs.

Je remercie aussi mes examinateurs Yulia, Vincent et Emmanuel pour avoir accepté de participer à ma soutenance.

Je remercie aussi tous les chercheurs qui m'ont beaucoup aidé lors de mon travail, qui m'ont encouragé dans mes idées et qui m'ont aidé pour mes enseignements :

César, Jean-Charles, Vincent, Nazim, Sylvain, Yann, Alain et Jérémy.

Je tiens ensuite à remercier toutes les personnes qui m'ont inspiré et avec qui j'ai partagé de nombreuses conversations constructives et divertissantes :Georgios, Maxime, Mihai, Thomas, Matthieu, Nicole, Pedro, Cecilia, Jiri, Inaki et Adrien.

Je remercie aussi bien sûr ma famille et mes amis qui m'ont toujours encouragé dans mes entreprises et qui ont toujours cru en moi.

Enfin je remercie Masha qui m'a soutenu dans mon optimisme et dans mes doutes.

Sommaire

Table des figures

Introduction générale

Depuis la première architecture proposée par von Neumann l'avènement des calculateurs est indiscutable. On peut y voir une deuxième révolution industrielle bouleversant la majorité des occupations humaines.

Gordon Moore a postulé en 1965 que la complexité des semi-conducteurs, composants à la base des architectures matérielles des calculateurs, doublerait tous les ans puis, en 1975 il postula que le nombre de transistors dans un microprocesseur doublerait touts les deux ans.

Cette prédiction s'est révélée exacte jusqu'à présent, mais il a fallu accélérer les vitesses d'horloge jusqu'aux vitesses critiques où le bruit n'est plus supportable et il a fallu aussi paralléliser les architectures en introduisant de plus en plus de cœurs de calculs avec des caches associés et des mémoires partagées. Cette approche semble désormais trouver ses limites en termes de synchronisation, de coût et de consommation.

Maintenant que l'on se rapproche du «mur» au delà duquel il ne sera plus possible de continuer à profiter de cette course à la miniaturisation, l'heure est à la recherche de solutions alternatives.

L'avènement des moyens de calculs non conventionnels

En parallèle, des efforts de recherche importants sont faits dans des moyens de calculs non conventionnels pouvant plus ou moins se substituer au fonctions courantes des processeurs.

On peut séparer ces recherches en deux classes, celles qui cherchent à reproduire les calculs traditionnels avec des moyens techniques différents et celles qui cherchent à faire des calculs au sens général du terme, sans chercher à reproduire les paradigmes de calcul actuels. Ces dernières sont les plus intéressantes mais sont souvent appliquées à un domaine bien précis sans parvenir à révolutionner l'industrie pour l'instant. On peut citer le calcul quantique, le calcul neuromimétique, etc.

Les enjeux sont énormes et les objectifs ambitieux : est-il possible d'accélérer le calcul des problèmes complexes (résolution de l'explosion combinatoire avec l'ordinateur quantique) ? Peut-on obtenir un apprentissage automatique aussi (plus) performant que l'humain avec le calcul neuronal et l'ingénérie neuro-mimétique ? Quelle résistance aux fautes et quelle robustesse permet d'atteindre le calcul spatial émergent ? L'ambition est donc de trouver des moyens de calculs beaucoup plus performants mais surtout beaucoup plus intelligents et adaptables que les machines actuelles. On peut noter que les meilleurs exemples d'inspiration pour ces types de calculs sont souvent trouvés dans la nature.

L'inspiration de la nature

La nature, et surtout la biologie, est une source constante de fascination pour un ingénieur. Les processus impliqués sont robustes, efficaces, adaptables et performants d'un point de vue

énergétique. Il est donc tentant de s'inspirer des mécanismes biologiques afin d'en extraire les grands principes d'apprentissage, d'évolution, d'auto-organisation, d'émergence, etc. Certains de ces principes ont déjà contribué à des algorithmes à succès comme l'optimisation évolutionnaire, les réseaux de neurones, les cartes auto-organisatrices. Mais la recherche souhaite aller plus loin en s'inspirant aussi des mécanismes physiques de ces calculs biologiques afin de créer de nouveaux substrats de calculs.

Certains projets sont atypiques comme les ordinateurs chimiques, les ordinateurs biologiques à base d'ADN, de molécules ou bien de micro-organismes (de la moisissure par exemple). Mais il y a pour l'instant peu d'applications industrielles.

Néanmoins on peut lister les propriétés du vivant plus ou moins formalisées et qui sont désirables pour nos substrats matériels. La liste de ces propriétés est longue et très contraignante mais elle résume des décennies de recherche scientifique et philosophique sur la définition d'un organisme vivant. Dans la perspective de créer une machine aussi performante que le vivant, toutes les propriétés suivantes peuvent apparaître souhaitables.

Propriétés des substrats physiques Le substrat physique sur lequel s'est développé la vie a des propriétés fondamentales qui conditionnent l'émergence de la vie telle qu'elle est. Au delà de toutes les lois chimiques et physiques on peut donner quelques grands principes à retenir.

Principe de localité : la communication d'information est locale car la transmission d'information requiers un transport de particule. C'est une condition très forte qui va inspirer le calcul cellulaire.

Invariance par symétrie, rotation et réflexion des réactions physico-chimiques. Ce principe de symétrie est très important puisqu'il permet une régularité et une invariance des substrats.

Réversibilité des processus physiques. Un processus est dit réversible si il n'y a pas d'augmentation de l'entropie. On peut donc inverser l'axe temporel pour obtenir le processus inverse. Cette propriété est la source d'actives recherches car elle permet une économie d'énergie dans les calculs. Et enfin la conservation des éléments, propriété de base des systèmes chimiques.

Il est intéressant de se questionner sur les principes physico-chimiques nécessaires à l'émergence de la vie et donc à quel point le substrat de simulation doit être proche de la nature. Une autre question importante pour le substrat de simulation est sa nature continue ou discrète. D'après Toffoli les comportements macroscopiques continus émergent d'un mécanisme microscopique discret. Il est donc en théorie possible d'utiliser un média discret.

La régénération et l'auto-organisation Les processus de régénération et d'auto-organisation sont à la base des mécanismes de survie des êtres vivants. C'est l'activité principale des êtres vivants qui puisent des ressources dans l'environnement afin d'en extraire les nutriments nécessaires à leur survie. Cette capacité est très intéressante pour un ingénieur vu qu'elle assure une tolérance très forte aux fautes et aux agressions de l'environnement.

L'auto-reproduction John von Neuman dans son article de 1948 «Théorie générale et logique des automates» pose la question de savoir s'il est possible pour une machine de s'auto-reproduire. La biologie montre qu'il est nécessaire de séparer la phase de construction et la phase de reproduction. Une cellule va d'abord fabriquer les protéines nécessaires à sa reproduction grâce à la transcription de l'ADN en ARN puis la traduction de l'ARN en protéine. Von Neumann était arrivé à la même conclusion avant même que l'ADN soit connu en proposant un automate cellulaire qui séparait sa reproduction en construction puis duplication du plan de construction.

L'évolution L'évolution est aussi beaucoup étudiée car c'est par ce mécanisme émergent que la vie s'est développée et s'est suffisamment complexifiée pour atteindre les niveaux d'intelligence que l'on souhaite pour nos machines. De nombreux modèles existent plus ou moins proches de la complexité biologique, le plus connu étant les algorithmes génétiques largement utilisés dans des contextes d'optimisation méta-heuristique.

Il a été montré qu'une telle évolution était capable d'optimiser la cognition et la structure de créatures artificielles. Un algorithme populaire pour l'évolution de la cognition est NEAT (neuroevolution of augmenting topologies) qui permet de développer la structure d'un réseau de neurones en fonction de la complexité de la tâche.

La morphogenèse La morphogenèse est définie par Alan Turing dans son célèbre papier «The chemical basis of morphogenesis». Dans ce papier il montre comment des instabilités dans des systèmes chimiques de réaction-diffusion (des morphogènes) sont à l'origine de motifs complexes dont certains sont observables sur la surface d'êtres vivants.

Ces résultats ont des conséquences très intéressantes puisqu'ils montrent comment la structure complexe des êtres vivants est une propriété émergente des interactions chimiques des morphogènes régulant l'expression génétique des cellules (différenciation).

La plasticité corticale Le cortex forme un ensemble de neurones et de synapses robuste, adaptable et capable de calculs complexes. C'est donc l'une des principales inspirations pour le calcul bio-inspiré. Ces propriétés sont majoritairement dues à la plasticité neuronale qui permet l'apprentissage, la réorganisation des ressources en cas de lésion, et même la régénération par la neurogenèse. Ceci permet aux êtres vivant dits «intelligents» de s'adapter à l'environnement de façon efficace et rapide.

Les calculs possibles dans le cortex sont de nature très différente des calculs possibles dans nos calculateurs. Le type de tâche auxquelles un ordinateur excelle sont le calcul et la mémorisation. Le cortex est lui spécialisé dans l'intégration d'informations sensorielles bruitées pour en extraire les informations essentielles dans une situation donnée et agir en conséquence.

Les approches de robotique et d'intelligence artificielle visent à donner aux machines la capacité de s'adapter à un environnement où à des tâches différentes et parfois imprévues grâce à des capteurs et actionneurs bruités. L'approche inspirée du cortex prend donc tout son sens.

Toutes ces propriétés sont souhaitables pour un substrat de calcul robuste et intelligent. De nombreux domaines de recherche ont étudié ces propriétés et un outil privilégié pour ça est l'automate cellulaire (AC). Ainsi von Neuman a pu étudier le problème de l'auto-reproduction grâce à un modèle simplifié du vivant sur les conseils de Stanislaw Ulam. Depuis, de nombreux résultats ont été obtenus qui ont permis de donner un cadre théorique à quelques-uns des concepts ci-dessus mais on a pu aussi prouver l'universalité de certains automates et raffiner les concepts d'émergence.

Ainsi on trouve des AC auto-reproducteurs (von Neumann, langton), des automates réversibles (Margolus), des automates à conservation de matière (HPP), et certains automates auto-reproducteurs ont été modifiés afin de faire émerger une évolution (sexy loops). Des architectures plus complexes ont été proposées dans le cadre des recherches sur le matériel bio-inspiré avec des capacités évolutives, de réparation, etc. En enlevant certaines contraintes que l'on a avec les AC afin d'obtenir une architecture avec toutes les propriétés voulues. Malheureusement ces systèmes peuvent difficilement atteindre les échelles voulues par la complexité des mécanismes engagés.

Le parallélisme massif comme substrat de calcul

Outre les qualités des automates cellulaires comme outil de modélisation, l'avantage du calcul cellulaire quel qu'il soit est la facilité de son implémentation dans des architectures matérielles traditionnelles. On bénéficie en effet de la simplicité relative de chaque cellule et du fait que les cellules sont similaires et réparties homogènement sur une grille à la topologie généralement simple. Et la connectivité est généralement limitée à un nombre de voisins restreint. Cette homogénéité et localité du substrat a pour désavantage de restreindre les communications à longue distance. La solution adoptée par les architectes est d'enlever certaines contraintes des automates cellulaires en utilisant un réseau sur puce. Ce réseau est souvent complexe et limité en termes de bande passante. Une autre solution est de diffuser l'information de proche en proche de façon à ce qu'elle atteigne sa destination.

Compte tenu de ses différentes propriétés, et notamment de sa capacité à répondre aux enjeux de robustesse et de mise à l'échelle, nous avons fait le choix de mettre le calcul cellulaire au centre de cette thèse. Dès lors se pose la question du niveau de modélisation. Faut-il modéliser au niveau des molécules, au niveau des cellules, ou au niveau des organismes ? La réponse n'est pas évidente d'autant plus qu'en modélisant à un niveau supérieur, on peut facilement perdre des propriétés intéressantes comme la symétrie ou la localité. L'idéal serait de réussir à reproduire la hiérarchie des émergences qui permettent l'apparition d'êtres intelligents mais il faudrait dans ce cas partir du substrat de base qui est nanoscopique et donc très difficilement manipulable.

À long terme le but des ces recherches est de parvenir à reproduire l'intelligence et la robustesse trouvées dans les créatures animales. Il est cependant difficile de faire émerger un calcul intelligent dans un substrat cellulaire. Même si certains automates cellulaires sont Turing-complet, cette propriété demande une très grande échelle de calcul et est peu robuste aux fautes matérielles.

C'est pourquoi il apparaît convenable de se placer à l'échelle de modélisation des neurones et des synapses. Les interactions synaptiques étant très denses et complexes nous nous placerons même à un niveau de modélisation supérieur, le niveau mésoscopique des populations de neurones. On peut ainsi abstraire la complexité des interactions synaptiques et se concentrer sur le comportement émergent d'une population entière de neurones. Nous verrons dans la progression de ce manuscrit que ce niveau de modélisation permet déjà beaucoup d'applications en calcul embarqué.

Malgré ce niveau de modélisation assez élevé il est quand même souhaitable de respecter un maximum de principes du calcul cellulaire afin d'avoir un substrat de calcul robuste.C'est le défi posé par cette thèse. Est-il possible d'obtenir les calculs émergents des populations de neurones avec un substrat numérique cellulaire ?

Nous verrons dans le premier chapitre les propriétés fondamentales du substrat cortical et comment sa dynamique est modélisée mathématiquement afin d'en extraire les grands principes de calculs. Nous décrirons alors la théorie des champs neuronaux dynamiques (Dynamic Neural Fields, DNF en anglais), modèle mésoscopique à la base de nos architectures. Nous montrerons alors comment ces equations sont non seulement capables de modéliser des résultats expérimentaux sur la dynamique corticale mais comment elles peuvent aussi être utilisées dans un cadre de robotique comportementale cortico-inspirée.

Nous étudierons ensuite dans un second chapitre comment les modèles neuronaux peuvent être implémentés dans un substrat matériel numérique tel que les Field Programmable Gate Arrays (FPGA). Ces cartes matérielles reconfigurables seront la cible privilégiée de nos architec-

tures pour leur facilité d'utilisation pour le prototypage. Nous nous intéresserons principalement aux architectures parallèles qui sont les plus prometteuses en termes de robustesse et qui ont l'avantage de se différencier des architectures de von Neumann. Nous proposerons alors deux architectures parallèles basées sur les principes du calcul neuro-mimétique permettant de simuler les champs neuronaux dynamiques de façon intégrée avec des capteurs bio-inspirés basés sur les événements tel que les rétines artificielles impulsionnelles.

Les troisième et quatrième chapitres présentent deux architectures originales permettant de simuler le calcul des DNF sur un substrat cellulaire. Le premier modèle, RSDNF pour Randomly Spiking Dynamic Neural Fields ou champs neuronaux dynamiques impulsionnels aléatoires, simule la propagation des informations latérales des neurones avec une transmission aléatoire de bits de proche en proche. Nous montrerons alors comment ce modèle décentralisé local (chaque unité de calcul est connectée à ses quatre voisins) permet de reproduire la dynamique des DNF.

Le deuxième modèle, CASAS-DNF pour Cellular Array of Stochastic Asynchronous Spiking ou Grille cellulaire de DNF impulsionnels aléatoires et asynchrones, améliore le modèle précédent afin de le rendre compatible avec un calcul asynchrone.

Pour tous ces modèles nous évaluerons leur capacité de mise à l'échelle, leur robustesse et leur dynamique émergente. Nous verrons ainsi quels sont les avantages et les inconvénients pour le développement d'architectures matérielles mélangeant plusieurs niveaux de modélisation afin de profiter des capacités de calcul du niveau cortical et de la robustesse et la facilité d'implémentation du niveau cellulaire.

Chapitre 1

Les champs neuronaux dynamiques

Les champs neuronaux dynamiques (DNF pour Dynamic Neural Fields en anglais) ont été introduits par Wilson et Cowan en 1972 [Wilson and Cowan, 1972] sous la forme d'une équation aux dérivées partielles (PDE pour Partial Differential Equation en anglais). La motivation principale était d'avoir un modèle macroscopique des populations de neurones qui soit suffisamment générique pour modéliser les dynamiques variées du cerveau. Le modèle a ensuite été développé par Amari [Amari, 1977] et étendu en deux dimensions par Taylor [Taylor, 1999].

Ce modèle s'est révélé suffisamment simple pour en extraire analytiquement des résultats sur les bifurcations et la convergence des PDE et pour modéliser ainsi des résultats physiologiques. Le modèle a connu ensuite beaucoup d'extensions afin d'améliorer son réalisme et son pouvoir de modélisation.

Comme nous le verrons dans cette partie, il est possible de simuler le comportement des DNF en discrétisant spatialement les PDE en une carte de neurones qui exécutent tous la même equation et qui interagissent. Par conséquent nous pouvons soutenir que la dynamique de la version spatialement discrète des DNF *émerge* de l'interaction d'unités simples de façon décentralisée.

Cela fait donc de ce modèle un candidat idéal pour la réalisation de calculs complexes émergents et décentralisés qui est la motivation de cette thèse.

Le modèle de Wilson et Cowan a beaucoup évolué dans les dernières décennies et est utilisé sous différentes appellations en fonction du domaine d'application. Le terme *neural field* a été introduit par Amari dans [Amari, 1977]. Ensuite, le terme de DNF a été introduit par Schöner indifféremment sous la forme *champs neuronaux dynamiques* ou *champs dynamiques* (dynamic fields) en 1995 dans [Schöner et al., 1995]. Il est aussi utilisé par [Erlhagen and Bicho, 2006, Sandamirskaya, 2013, Bicho et al., 2010, Johnson et al., 2009, Fix et al., 2011b, Rougier and Vitay, 2006]. Les architectures de robotique autonome bio-inspirées ont été regroupées dans le cadre de la théorie des champs dynamiques (DFT pour Dynamic Fields Theory en anglais) introduite par [Erlhagen and Schöner, 2002] et utilisée dans de nombreuses applications [Wilimzig et al., 2006, Zibner et al., 2010, Faubel and Schöner, 2008, Sandamirskaya and Schoner, 2008, Simmering et al., 2008].

Les DNF sont aussi utilisés en version discrète ou continue dans les réseaux de neurones à attracteurs continus (CANN pour Continuous Attractor Neural Networks en anglais). Ces travaux sont issus de [Samsonovich and McNaughton, 1997] qui se base sur les travaux d'Amari, d'Amit et de Tsodyks [Amit, 1992].
La motivation initiale était différente. Le but de ce modèle était de modéliser les cellules de la direction de la tête chez le rat (Head Cells) [Zhang, 1996, Taube and Bassett, 2003].

Les termes *attracteurs à bosse* (bump attractors en anglais) ou *attracteurs gaussiens* sont aussi utilisés [Deneve et al., 2001, Wimmer et al., 2014] avec des applications (voir par exemple [Hu and Zhang, 2010, Fung and Amari, 2015]).

Malgré l'engouement pour ce modèle il n'y a pas vraiment d'étude pour une architecture matérielle dédiée à la simulation des DNF ou des CANN, à l'exception d'une architecture neuromorphique sur VLSI (Very Large Scale Integration) étudiée pour la simulation de CANN uni-dimensionnel dans le cadre de la simulation des cellules de la direction de la tête découvertes chez le rat [Massoud and Horiuchi, 2011]. Dans [Giese, 2012] il y a aussi une proposition intéressante pour implémenter les champs neuronaux dynamiques à l'aide d'un réseau de neurones cellulaire (CNN pour Cellular Neural Network en anglais) sur VLSI. L'idée serait de s'inspirer des filtres de Gabor spatio-temporels qui ont une dynamique similaire et dont l'implémentation sur CNN est proposée dans [Shi, 1996]. Bien que l'idée soit intéressante nous ne l'explorerons pas car cette thèse est consacrée à des substrats numériques.

Nous décrirons dans cette partie le modèle original des DNF ainsi que les extensions principales. Nous en verrons ensuite la dynamique et les applications.

1.1 La théorie des champs neuronaux continus

Dans ce chapitre nous allons discuter des bases biologiques des champs neuronaux dynamiques. Ce n'est pas fondamental pour la progression du manuscrit mais c'est un bon exemple pour comprendre comment l'on peut extraire des principes généraux d'un système plutôt que de chercher à avoir une vue de plus en plus fine des éléments du système. De plus, certaines extensions des DNF sont directement inspirées de phénomènes neuronaux ou synaptiques. Il est donc nécessaire d'avoir des connaissances de base pour pouvoir en comprendre la finalité.

La principale motivation de la théorie des champs neuronaux continus est de pouvoir prédire le comportement statistique d'une population de neurones, plutôt que de se concentrer sur la dynamique d'un seul neurone qui est beaucoup trop bruitée pour en extraire une information fiable [London et al., 2010]. Après avoir décrit la dynamique des neurones et des synapses nous allons décrire l'organisation du cortex et comment celle-ci a inspiré la théorie des champs neuronaux dynamiques.

1.1.1 Bases biologiques

Les champs neuronaux dynamiques sont le fruit d'une recherche qui s'est d'abord concentrée sur la description des unités fondamentales du cerveau (neurones, synapses) puis sur la dynamique et la plasticité de ces unités avant de prendre un peu de hauteur pour décrire les zones physiologiques et fonctionnelles du cerveau. Ce n'est qu'à partir de ces résultats qu'il a été possible de proposer un modèle crédible de population neuronale.

1.1.1.1 Neurone

Le neurone est la cellule fondamentale de l'intégration et de la transmission des informations nerveuses. Le corps cellulaire d'un neurone typique est composé du noyau et des organites standards des cellules eucaryotes. La spécificité des neurones réside en leurs prolongements membranaires de deux types maintenus par un cytosquelette très dense. L'axone (ou fibre nerveuse) est un prolongement de diamètre 1 à 15 μm pouvant avoir un axe principal très long avant de se ramifier afin d'établir des connexions synaptiques avec les dendrites d'autres neurones. Les

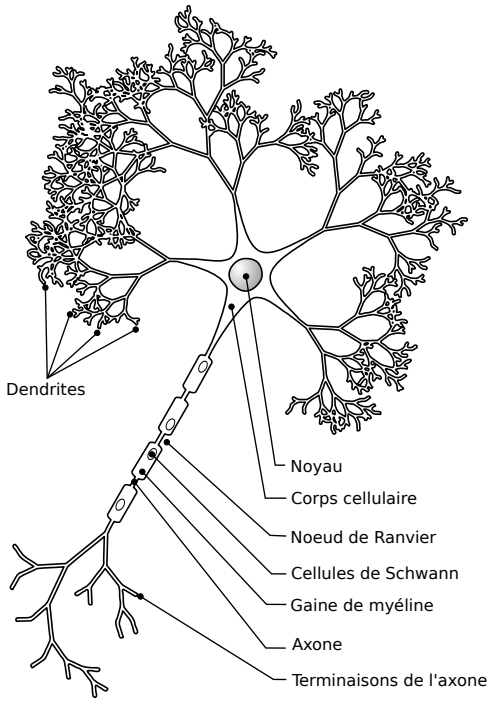


FIGURE 1.1 – Anatomie d'un neurone. Source : N.P. Rougier, CC BY-SA 3.0.

dendrites sont l'autre type de prolongement cellulaire qui sont caractérisés par une arborescence très dense mais courte comparée à l'axone (fig. 1.1).

1.1.1.2 Le potentiel d'action

Au repos le neurone maintient activement un potentiel de -70mv en expulsant activement les ions Na+ (sodium) et en absorbant activement les ions K+ (potassium) avec les pompes NaK sensibles à l'ATP [1].

Les canaux sodiques sont sensibles au voltage. C'est à dire que lorsque le potentiel membranaire est perturbé par la réception des potentiels post-synaptiques au niveau des dendrites, certain canaux sodiques s'ouvrent. Lorsque le potentiel membranaire atteint un seuil d'excitabilité de -50mv, tous les canaux sodiques s'ouvrent jusqu'à ce que le potentiel atteigne le potentiel d'équilibre du sodium : 60mv environ. Les canaux sodium s'inactivent ensuite progressivement par obturation.

Les canaux de rectification sensibles au voltage laissent sortir le potassium qui repolarise la membrane en suivant le gradient électrochimique. Cette rectification est trop importante ce qui mène à une hyper-polarisation de la membrane (fig 1.2).

Une période réfractaire absolue s'ensuit car de nombreux canaux sodiques ne sont pas encore inactivés. Puis une période réfractaire relative où le stimulus doit être très fort pour provoquer une dépolarisation.

Cette dépolarisation locale va provoquer la dépolarisation de la partie voisine de l'axone et ainsi le potentiel d'action va se propager jusqu'au bout de l'axone à la vitesse approximative de $1m.s^{-1}$

1. L'ATP (adénosine tri phosphate) est la molécule de « transaction énergétique» des cellules.

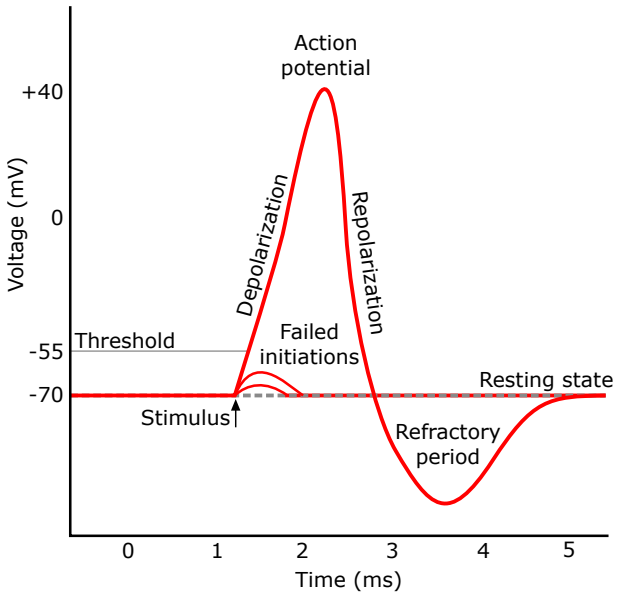


FIGURE 1.2 – Potentiel d'action. Source : Chris73, CC BY-SA 3.0

La propagation des potentiels d'action peut être accélérée sur les axones longs qui sont myélinisés et qui permettent une conduction à une vitesse de l'ordre de $100m.s^{-1}$

1.1.1.3 La synapse

Les connexions synaptiques assurent la transmission des potentiels d'action de l'axone d'un neurone à la dendrite d'un autre. La majorité des synapses sont chimiques et communiquent l'impulsion électrique par l'intermédiaire de neuro-transmetteurs.

Lorsqu'un potentiel d'action atteint l'extrémité de l'axone (ou élément pré-synaptique) le changement de polarité provoque un afflux de calcium qui provoque la fusion de vésicules pleines de neuro-transmetteurs avec la membrane cellulaire provoquant ainsi la libération des neuro-transmetteurs dans la fente synaptique.

Les neuro-transmetteurs se fixent alors sur les récepteurs membranaires de l'élément post-synaptique. Les récepteurs sont des protéines-canal qui s'ouvrent et génèrent un potentiel post-synaptique qui peut être excitateur ou inhibiteur. Le potentiel post synaptique excitateur réduit la différence de potentiel cellulaire du neurone alors que l'inhibiteur augmente cette différence.

Les dendrites ont peu de canaux sodiques, elle ne peuvent donc pas générer un potentiel d'action. En revanche les potentiels post-synaptiques se propageront jusqu'au pérycarion du neurone qui est une zone riche en canaux sodium et qui est la source des potentiels d'actions générés par l'axone.

Les neuro-transmetteurs les plus souvent impliqués dans les processus synaptiques sont le glutamate (qui a pour récepteurs AMPA, NMDA et kïnate) qui est excitateur, GABA (un métabolite du glutamate) qui est inhibiteur (il a pour récepteur $GABA_A$), et l'acétylcholine (excitateur) qui est impliqué dans la mémoire et l'apprentissage et dans le système nerveux autonome car il régule l'activité musculaire. Ses récepteurs sont les récepteurs nicotiniques et muscariniques.

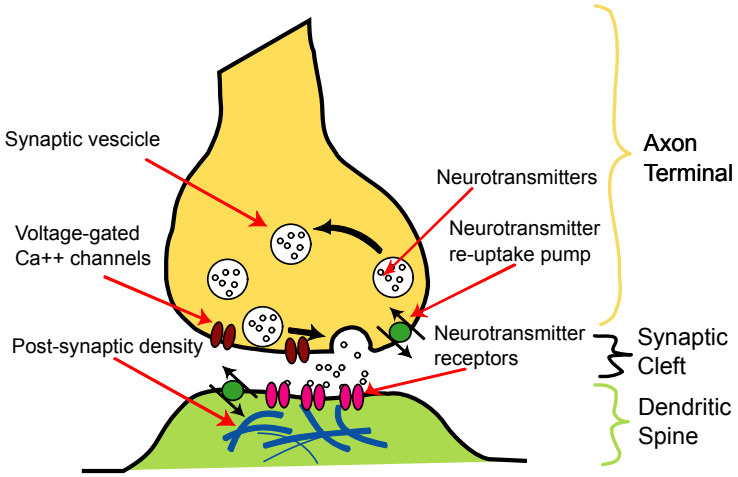


FIGURE 1.3 – Synapse. Source : Nrets, CC BY-SA 3.0

Les neurones connectés par des synapses sont donc les unités fondamentales d'intégration et de transmission des informations. Leur dynamique riche n'est pas encore complètement comprise et est le siège de régulations et modulations complexes au niveau membranaire comme synaptique. A cette complexité s'ajoute un bruit inhérent parfois difficile à quantifier et dont les sources sont variées. La variabilité des potentiels d'actions générés par un neurone soumis à un stimulus constant peut être de l'ordre de la milliseconde [Faisal et al., 2008]. Malgré le fait que ce bruit suit une loi aléatoire de Poisson les neuro-scientifiques ne s'accordent pas sur le fait que ce soit entièrement généré par un processus aléatoire. Shanon lui même a prouvé que lorsqu'un encodage est optimal, les signaux semblent aléatoires. Des preuves de plus en plus fortes supportent que le bruit fait partie intégrante de l'encodage neuronal bien qu'on ne soit pas encore sûr de savoir lequel.

Ces conclusions motivent donc la recherche de modèles à échelle macroscopique utilisant une version simplifiée des neurones. Il est pour cela utile d'observer la structure macroscopique du cortex.

1.1.1.4 Physiologie

Dans la démarche bio-inspirée qui nous motive, il est intéressant d'étudier l'organisation fonctionnelle du cortex afin d'en tirer des grand principes pouvant se révéler utiles à la modélisation voire au développement de matériel neuromorphique.

1.1.1.4.1 La surface corticale Le cortex est la couche externe du cerveau (violet foncé dans la figure 1.4). Comme le suggère cette photo, le cortex est organisé en feuillets.

En effet au cours de l'évolution, le cerveau n'a cessé d'augmenter son volume mais cette augmentation est essentiellement due à une augmentation de la surface plutôt que de son épaisseur [Wilson and Cowan, 1973]. Cela explique la structure fractale du cerveau. On peut par exemple comparer les caractéristiques des cerveaux du lapin et de l'homme. On passe d'une surface de $843mm^2$ à $83591mm^2$ alors que l'épaisseur passe de $1.74mm$ pour le lapin à $3mm$ pour l'homme. Le nombre de neurones augmente, leur volume et leurs ramifications aussi. On voit donc que la complexité de comportement chez les vertébrés s'accompagne d'un accroissement de l'épaisseur corticale mais surtout de la surface corticale.

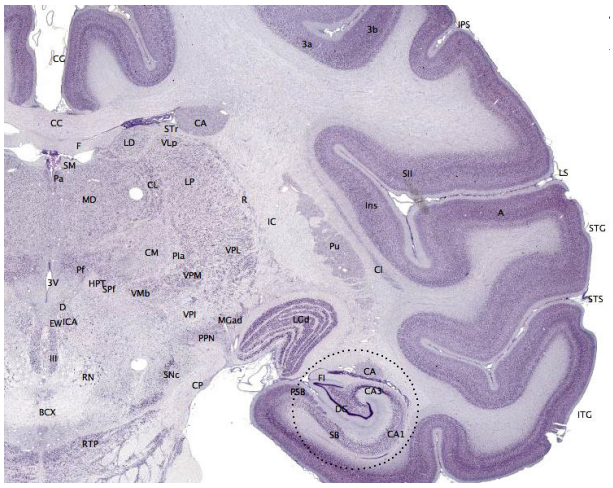


FIGURE 1.4 – Cortex cérébral. Source : brainmaps.org, CC BY 3.0

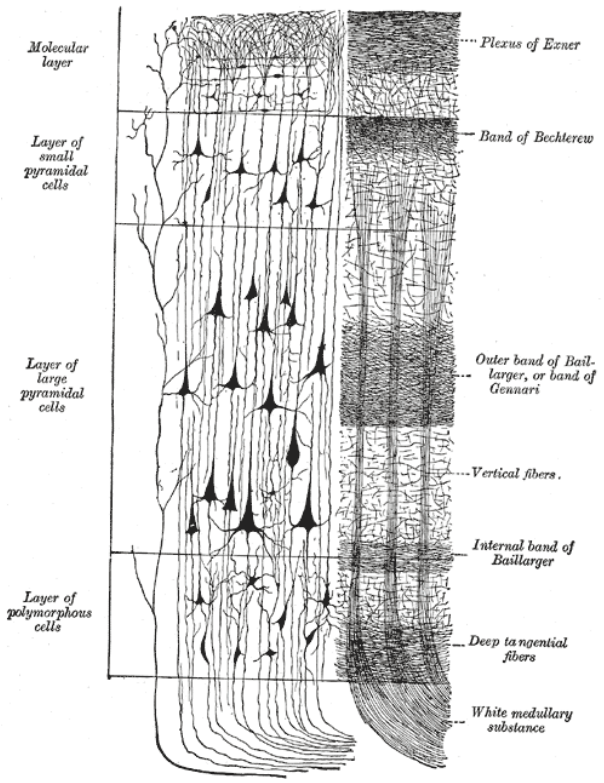


FIGURE 1.5 – Couches histologiques du cortex [y Cajal, 1888].

Une étude histologique donne des informations précises sur la structure fonctionnelle du cortex (voir fig. 1.5). Dans la description faite par Ramon Y Cajal on peut voir plusieurs couches correspondant à différentes fonctions [y Cajal, 1888].

1. La couche moléculaire contient des axones et des dendrites
2. La couche granulaire contient les neurones granulaires qui reçoivent les afférences des autres aires du cortex.
3. La couche pyramidale externe contient des cellules pyramidales et émet des connexions vers d'autre zones du cortex.
4. La couche granulaire interne contient des neurones étoilés et pyramidaux et reçoit les connexions en provenance de l'extérieur du cortex.
5. La couche pyramidale interne envoie des connexions vers l'extérieur du cortex (ex motoneurones).
6. La couche polymorphe envoie des connexions vers le thalamus.

1.1.1.4.2 Colonnes corticales Mountcastle a étudié la réponse des neurones du cortex somato-sensoriel aux déformations du tissu de la partie controlatérale du chat [Mountcastle, 1957]. Dans toutes les couches de cellules, il y a trois types de neurones : ceux qui répondent aux poils, ceux qui répondent à la pression sur la peau et ceux qui répondent aux déformations plus profondes des tissus. Ces neurones sont organisés en colonnes (micro-colonnes) et ces colonnes sont entremêlées sur l'ensemble des couches analysées. Chaque cellule à un champ récepteur assez large et ces champs récepteurs se recoupent. De même beaucoup de travaux ont étudié l'organisation en colonnes du cortex visuel et sont résumés par [Swindale, 1996]. Les cartes de dominances oculaires et des colonnes d'orientations montrent que les colonnes sont spécialisées de façon à représenter l'espace des stimuli.

Ces observations ont mené à la théorie de l'organisation en colonnes corticales du cortex : les micro-colonnes répondent à des stimuli similaires et les macro-colonnes contiennent un ensemble fonctionnel de micro-colonnes [Hubel and Wiesel, 1963]. Une micro-colonne contient environ 100 neurones et une macro-colonne 10000 neurones.

De plus, toutes les stimulations sensorielles sont projetées sur une surface bi-dimensionnelle. Cela montre donc que le cortex est organisé en aires fonctionnelles et qu'il peut être vu comme une hiérarchie complexe de ces aires [Wilson and Cowan, 1973].

1.1.1.4.3 Connectivité La connectivité au sein de ces colonnes corticales et entre les colonnes corticales est à la base de tous les modèles proposés. Le premier constat est la densité des connexions ; chaque neurone est connecté à plusieurs milliers d'autres neurones. Cette densité est une caractéristique essentielle des mécanismes neuronaux et, comme nous le verrons, celle qui donne le plus de difficultés aux implémentations matérielles. La densité des synapses est essentielle pour que l'apprentissage soit riche et profond. Les neurones sont séparés en neurones excitateurs et en neurones inhibiteurs et ont des connexions de même nature. La majorité des interactions inhibitrices à faible portée est due aux inter-neurones, neurones multipolaires assurant les connexions entres les réseaux afférents et efférents. La quantité importante d'informations accumulée dans la dernière décennie sur les inter-neurones inhibiteurs montre beaucoup de diversité. Le rôle de chaque inter-neurone n'est cependant pas encore précisément identifié [Wang et al., 2004]. La plupart des modèles supposent que les connexions sont réparties de telle sorte que l'excitation soit locale (connexions excitatrices courtes) alors que les inter-neurones

inhibiteurs ont des connexions plus longues permettant une inhibition plus large. Les preuves physiologiques sont encore débattues [Kang et al., 2003].

Il y a bien sûr des connexions dont la portée dépasse la simple colonne verticale et permettent des communications inter-aires corticales [Douglas and Martin, 2012]. Mais ces connexions font plutôt partie de l'organisation macroscopique du cortex qui n'est pas notre niveau de modélisation dans ce manuscrit.

On voit donc que même une description superficielle de la biologie du cerveau fait appel à de nombreux résultats issus de nombreuses années de recherche. Les neurones connectés par l'intermédiaire de synapses plastiques sont les composants principaux. Ces neurones sont ensuite organisés en micro-colonne (100 neurones) qui ont un champ récepteur quasi identique. Les micro-colonnes sont ensuite assemblées en macro-colonnes (10000 neurones) qui constituent un ensemble de neurones au champ récepteur multi-dimensionnel (orientation + dominance binoculaire par exemple). Ces colonnes sont disposées sur une structure bi-dimensionnelle.

Les différentes zones corticales sont connectées par des neurones afférents dont les axones projettent les informations nerveuses à longue distance.

Le fonctionnement du cerveau des vertébrés supérieurs est le fruit de nombreuses années d'évolution où les contraintes spatiales et comportementales expliquent la structure stratifiée du cortex. L'organisation des zones corticales est le fruit d'une croissance orientée par des morphogènes lors de la maturation du cerveau chez l'embryon (évolution ontogénique). Une grande partie de l'organisation est aussi le fruit de l'apprentissage commencé au même moment mais continué tout au long de la vie. La part des contributions innées, développementales ou apprises fait toujours débat et une modélisation précise de ces trois contributions est probablement nécessaire à l'explication de la relation structure/fonction des différentes aires du cerveau [Swindale, 1996].

Mais plutôt que de chercher à modéliser l'apprentissage de toutes ces fonctions il peut être utile de modéliser les fonctions directement grâce à des équations suffisamment génériques pour pouvoir décrire la plupart des mécanismes corticaux.

1.1.2 Description analytique

Les travaux en anatomie et en physiologie progressant, certaines tentatives ont été faites pour avoir une vue générique du comportement d'une population de neurones. Comme les particules qui ont un mouvement Brownien en physique, un neurone seul a une activité très bruitée qu'il est difficile de prédire. En revanche tout porte à croire qu'il y a une redondance locale des réponses des neurones aux stimuli. Par conséquent il semble qu'il soit plus facile de prédire la réponse d'une population locale de neurones plutôt que celle d'un neurone seul. Comme en dynamique des fluides, il est donc plus intéressant de capturer la dynamique d'une population de particules plutôt que la dynamique d'une seule particule qui ne permet pas de prédire grand chose.

Nous allons ici introduire les principales notions et notations nécessaires à la modélisation des champs neuronaux dynamiques.

1.1.2.1 Modèle de Cowan and Wilson

L'approximation de continuité est faite en se basant sur l'hypothèse que les neurones d'une population sont spatialement proches et que leur interconnexion est aléatoire mais assez dense pour que tous les neurones soit interconnectés. Soit directement, soit par l'intermédiaire de quelque neurones (1 ou 2).

Il est supposé que les neurones sont distribués uniformément et que leur connectivité latérale dépend de la distance uniquement.

Une autre supposition est que tous les processus nerveux dépendent de l'interaction de neurones excitateurs et inhibiteurs.

Commençons par formaliser les interconnexions. Le tissu est isotrope et homogène pour toutes les connexions inhibitrice-inhibitrices, inhibitrice-excitatrices, excitatrice-inhibitrices et excitatrice-excitatrices. Autrement dit, la connectivité entre deux neurones dépend uniquement de leur type excitateur ou inhibiteur et de la distance qui les sépare, mais pas de leur position exacte. Ainsi, la probabilité que deux neurones de types excitateurs ou inhibiteurs $j \in \{e, i\}$ et $j' \in \{e, i\}$ à distance d soient connectés est donnée par

$$\beta_{jj'}(d) = b_{jj'} e^{\frac{-|d|}{\sigma_{jj'}}} \tag{1.1}$$

Chez le chat la distance moyenne des interactions latérales est typiquement $50\mu m$ et la vitesse de conduction axonale est plusieurs mm par ms. Le retard sera donc de l'ordre de 0.01 ms, ce qui est faible comparé aux constantes temporelles des neurones τ de plusieurs ms. Il est donc raisonnable de dire que la vitesse de transmission est infinie. Nous allons décrire l'activité des neurones excitateurs E et inhibiteurs I sur un continuum spatio-temporel où x décrit la position spatiale et t le temps. On obtient donc une équation de l'évolution de la moyenne temporelle de l'activité de E et I : $\langle E \rangle$ et $\langle I \rangle$ respectivement.

$$\begin{aligned} \tau \frac{\partial}{\partial t} \langle E(x,t) \rangle = & \\ & - \langle E(x,t) \rangle + [1 - r_e \langle E(x,t) \rangle] \\ & \mathscr{S}_e[\alpha\tau[\varrho_e \left(\langle E(.,t) \rangle * \beta_{ee}(||.||)\right)(x) - \varrho_i \left(\langle I(.,t) \rangle * \beta_{ie}(||.||)\right)(x) \pm \langle P(x,t) \rangle]] \end{aligned} \tag{1.2}$$

et de même pour la population inhibitrice :

$$\begin{aligned} \tau \frac{\partial}{\partial t} \langle I(x,t) \rangle = & \\ & - \langle I(x,t) \rangle + [1 - r_i \langle I(x,t) \rangle] \\ & \mathscr{S}_i[\alpha\tau[\varrho_e \left(\langle E(.,t) \rangle * \beta_{ei}(||.||)\right)(x) - \varrho_i \left(\langle I(.,t) \rangle * \beta_{ii}(||.||)\right)(x) \pm \langle Q(x,t) \rangle]] \end{aligned} \tag{1.3}$$

Où r_e (resp. r_i) est la période réfractaire absolue des neurones excitateurs (resp. inhibiteurs), $\mathscr{S}_e[N]$ (resp. $\mathscr{S}_i[N]$) est la proportion de neurones excitateurs (resp. inhibiteurs) recevant une excitation supérieure ou égale au seuil d'activation par unité de temps en fonction de N (sigmoïde), α est l'amplitude maximale du potentiel membranaire post-synaptique, θ est la constante temporelle membranaire, ϱ_e (resp. ϱ_i) est la densité de surface des neurones excitateurs (resp. inhibiteurs) dans un tissu uni-dimensionnel homogène et isotrope, et P et Q sont respectivement les activations afférentes des neurones excitateurs et inhibiteurs.

L'étude de la dynamique d'un tel système va s'avérer plus aisée sous une forme simplifiée telle que celle d'Amari.

1.1.2.2 Formulation d'Amari

Amari introduit les champs neuronaux par une simplification des équations 1.2 et 1.3 en regroupant les termes excitateurs et inhibiteurs [Amari, 1977] :

$$\tau \frac{\partial u(x,t)}{\partial t} = -u + \int w(x-y) F[u(y)] dy + h + s(x,t) \tag{1.4}$$

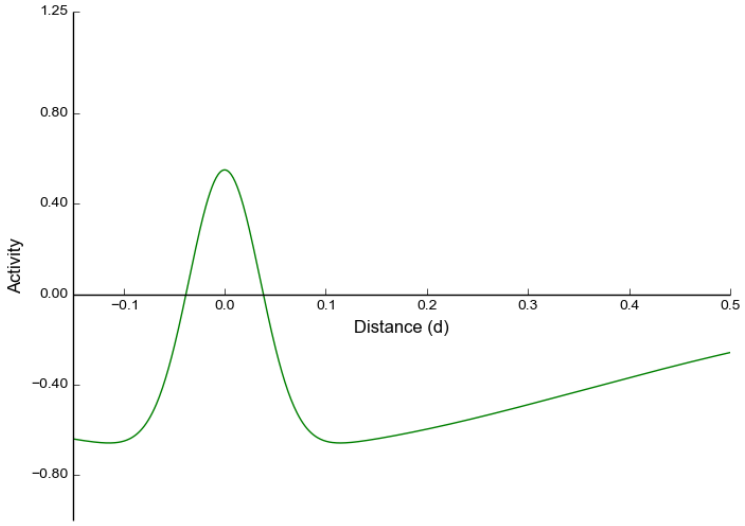


FIGURE 1.6 – Différence de gaussiennes

u est le potentiel membranaire du neurone, F est une fonction d'activation non linéaire. $w(|x-y|)$ est l'intensité moyenne de connexion du neurone situé à l'endroit x au neurone situé à l'endroit y et dépend uniquement de la distance $|x-y|$, $s(x,t)$ est l'activité afférente ou entrée du champ neuronal, h est le potentiel de repos.

La dynamique de ces champs neuronaux émerge d'une coopération locale et d'une compétition globale qui est donnée par la forme de la fonction des poids latéraux w en différence de gaussiennes ou chapeau Mexicain (voir fig. 1.6).

On voit donc que la formulation d'Amari est plus concise. Cependant elle ignore les neurones qui sont en période réfractaire.

L'extension en deux dimensions a été proposée et analysée par Taylor en 1999 [Taylor, 1999] :

$$\tau_i \frac{\partial u_i(x,y,t)}{\partial t} = -u + \int w(||x-x', y-y'||) F[u(x',y',t)] dx' dy' + h + s(x,y,t) \tag{1.5}$$

Où $||x-x', y-y'||$ est la distance euclidienne entre les positions (x,y) et (x',y').

Le modèle d'Amari est très simplifié puisqu'il ne prend pas en compte les délais synaptiques, l'apprentissage synaptique et fait beaucoup d'hypothèses sur la connectivité des neurones dans les colonnes corticales. De nombreux modèles dérivent de celui d'Amari et nous en étudierons certains dans la progression de ce manuscrit.

Cette section a permis d'introduire les concepts fondamentaux de la théorie des champs neuronaux continus. Nous avons d'abord vu la biologie simplifiée des neurones et des synapses. Nous avons vu qu'il existait des synapses excitatrices et inhibitrices ainsi que les neuro-transmetteurs impliqués dans ces mécanismes. Nous avons ensuite vu l'organisation de ces neurones en feuillets

divisés en micro- et macro-colonnes corticales et nous avons vu que la connectivité au sein de ces colonnes était dense entre les inter-neurones alors que les neurones afférents ont des axones plus long pour interconnecter les différentes aires cérébrales et les différentes couches de ces aires.

Cette vision simplifiée de la dynamique du cortex a permis à Wilson et Cowan de développer une équation différentielle partielle permettant de rendre compte de la dynamique des populations de neurones excitateurs et inhibiteurs au sein de ces macro-colonnes. Le modèle a ensuite été simplifié par Amari et étendu en deux dimensions par Taylor menant au modèle qui est utilisé dans cette thèse sous le nom de champs neuronaux continus.

Nous allons voir dans le prochain chapitre la dynamique de ce modèle pour la modélisation de processus physiologiques ou pour des applications en robotique autonome.

1.2 Étude dynamique et applications

La théorie des champs neuronaux continus a eu beaucoup de succès pour la modélisation des processus cognitifs hauts niveau et des dynamiques physiologiques du cerveau. Les PDE sont minimalistes et assez faciles à manipuler. Il est donc possible de prédire des résultats de convergence et de bifurcation en fonction de la forme des poids latéraux, de l'activité afférente, et de la dynamique des neurones et des synapses. Les premiers résultats analytiques se sont concentrés sur la dynamique d'une carte uni-dimensionnelle ou bi-dimensionnelle, identifiant ainsi des ondes progressives (travelling waves) et des solutions en bosse (bump solutions). Cela permet d'expliquer facilement des phénomènes d'intérêt médical comme l'épilepsie ou d'expliquer des mécanismes d'intérêt cognitif comme la représentation de la direction de la tête [Taube and Bassett, 2003], l'encodage des stimuli visuels et la représentation des mémoires de travail [Amari, 1977].

Ce chapitre sera organisé en deux parties, la première s'intéressera à la dynamique d'une seule carte neuronale alors que la seconde s'intéressera aux architectures multi-cartes.

1.2.1 Dynamiques mono-carte et extensions

Nous allons nous intéresser principalement aux dynamiques d'intérêt biologique ou robotique plutôt qu'à une description extensive des différentes solutions possibles telle qu'elle a été faite par Amari et Taylor et plus récemment par Coombes et Bressloff. Nous présenterons des résultats analytiques aussi bien que des résultats de simulation.

1.2.1.1 Résultats analytiques

Cette section apportera certains résultats analytiques (solution des PDE) et décrira surtout les paramètres ou les modifications du modèle nécessaires pour engendrer les dynamiques les plus remarquables. Cela permet aussi de se familiariser avec la dynamique de ce type de modèle.

1.2.1.1.1 Ondes progressives Les ondes progressives sont une propagation de l'activité électrique généralement observée lorsqu'une tranche de tissu cérébral est plongée dans un liquide pharmacologique bloquant l'inhibition $GABA_A$ [Golomb and Amitai, 1997]. Une stimulation locale d'une amplitude limite se propagera alors à une vitesse de 60 à 90 mm s^{-1}. Des enregistrements intracellulaires montrent qu'une telle vague correspond à une dépolarisation du potentiel membranaire qui conduit à une décharge à haute fréquence des neurones.

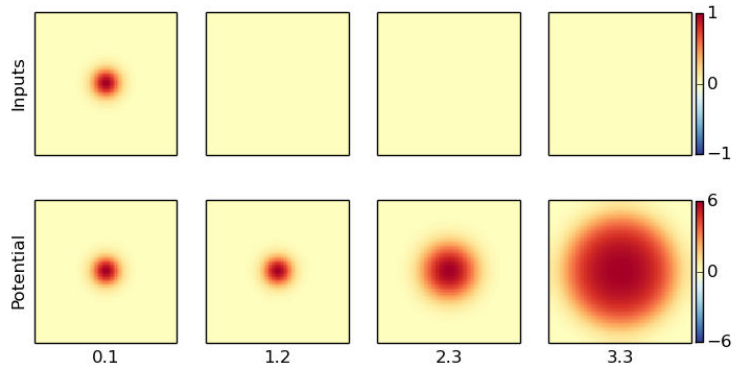


FIGURE 1.7 – Fronts progressifs (bas) après une impulsion de $1.2s$ (haut).

L'étude de ces ondes est intéressante car les différentes vitesses de propagation reflètent des différences anatomiques et physiologiques intéressantes dans les différentes aires du cerveau. Mais cela permet aussi d'étudier les conditions d'initiation de l'épilepsie.

Fronts progressifs Il est facile d'obtenir un front progressif d'activation avec un champ neuronal proposé par Amari. Il suffit d'enlever la partie inhibitrice des poids latéraux. On peut donc écrire les poids latéraux avec :

$$\omega(x) = \frac{1}{2\sigma} e^{-|x|/\sigma} \tag{1.6}$$

Cependant ce modèle n'est pas réaliste vu que dans un substrat biologique, les neurones ne restent pas activés indéfiniment. Il faut alors étendre le modèle d'Amari et introduire les champs neuronaux adaptatifs.

Impulsions progressives Même en l'absence d'inhibition synaptique les neurones ont un mécanisme de rétroaction inhibitrice. Les mécanismes candidats sont la dépression synaptique ou l'adaptation à la fréquence d'impulsion (SFA pour Spike Frequency Adaptation en anglais). Cette adaptation serait provoquée par un contrôle de courant potassique par le calcium et par le courant-M qui est un courant potassique sensible au voltage [Madison and Nicoll, 1984]. Ce courant d'hyper-polarisation a une constante temporelle de 40 à 120 ms. Il est possible de modéliser ce courant dans un modèle à fréquence en ajoutant à l'équation des neurones un courant négatif $q(x,t)$ [Benda and Herz, 2003]. La dynamique de ce courant dépend du potentiel u_i du neurone i avec :

$$\tau \frac{dq_i}{dt} = -q_i(t) + \gamma_q F_i(u_i(t) - q_i(t)). \tag{1.7}$$

Pinto et Ermentrout on proposé une simplification linéaire et spatialement continue de l'adaptation [Pinto and Ermentrout, 2001] :

$$\tau \frac{\partial q(x,t)}{\partial t} = -q(x,t) + \beta u(x,t) \tag{1.8}$$

où β est l'amplitude de l'adaptation et ϵ le τ la constante temporelle.

Ce modèle a une dynamique complexe pouvant mener à une variété d'activité (voir [Pinto and Ermentrout, 2001] pour les détails).

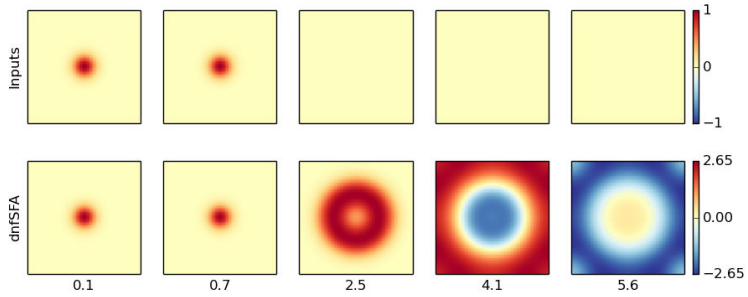


FIGURE 1.8 – Vague progressive. Une impulsion de 1s (haut) provoque une vague d'excitation suivie d'une vague d'inhibition dans les potentiels membranaires des neurones (bas).

Oscillations structurées En plongeant des feuillets néo-corticaux ou de l'hippocampe dans de la bicuculline qui inhibe les neuro-transmetteurs inhibiteurs il est possible d'observer une activité oscillante entre 1 et 10 Hz dans une population stimulée localement par un courant. L'activité générée peut se propager sous la forme d'impulsions progressives ou de vagues en spirale.

Le modèle précédent est capable de simuler des oscillations et des vagues en spirale corrélées aux données expérimentales. Mais le réalisme des paramètres d'adaptations étant douteux un autre modèle à été introduit faisant cette fois appel à un autre type d'adaptation, la dépression synaptique :

$$\frac{\partial u(r,t)}{\partial t} = -u(r,t) + \int w(|r-r'|)q(r',t)F(u(r',t))dr' \tag{1.9}$$

$$\frac{\partial q(r,t)}{\partial t} = \frac{1-q(r,t)}{\tau_q} - \beta q(r,t)F(u(r,t)). \tag{1.10}$$

1.2.1.1.2 Bulles d'activité Les bulles d'activité représentent le fonctionnement normal d'un champ neuronal en présence d'excitation et d'inhibition. Nous nous intéresserons dans cette partie aux bulles persistantes spontanées qui peuvent servir de mémoire de travail et aux bulles générées par des stimuli dans le cadre des mécanismes d'orientation de la tête.

Bulles d'activité localisées persistantes Les bulles d'activité localisées et persistantes ont été décrites par Amari [Amari, 1977] dans un contexte uni-dimensionnel. Il fait déjà le parallèle avec les mémoires de travail. La position de la bulle encoderait une information locale qui serait ensuite effacée grâce à une inhibition afférente.

Il est intéressant de voir que Amari a défini certaines propriétés pour la fonction de poids synaptique en forme de chapeau mexicain utilisée dans l'équation 1.4 [Bressloff, 2012] :

1. $w(-x) = w(x)$
2. $w(x) > 0, \forall x \in [0, x0)$ avec $w(x_0) = 0$
3. $w(x) < 0, \forall x \in (x_0, \infty)$
4. $w(x)$ est décroissant sur $[0, x_0]$

5. $w(x)$ a un seul minimum sur R^+ à $x = x_1$ avec $x_1 > x_0$ et $w(x)$ strictement croissant sur (x_1, ∞)

On peut donc s'y référer pour les modifications que l'on fait dans le cadre de l'implémentation matérielle.

Dans le cas mono-dimensionnel, on utilise l'équation d'Amari (eq. 1.4) avec une fonction Heaviside d'activation : $F(u) = H(u - \theta)$. θ est le seuil de la fonction Heaviside utilisée.

On modélise la solution avec :

$$U(x) = \int_{-\infty}^{+\infty} w(x - x')H[U(x') - \theta]dx' + h \tag{1.11}$$

On note alors $R[U] = \{x|U(x) > \theta\}$ la région du champ qui est excitée. On peut donc écrire la solution recherchée avec :

$$U(x) = \int_{R[U]} w(x - x')dx' + h \tag{1.12}$$

On peut arbitrairement appliquer une translation sur la solution pour la centrer. Pour identifier les conditions d'existence d'une bulle d'activité stable, on va supposer l'existence d'une telle bulle. On définit donc la solution de taille Δ comme la solution qui est activée sur l'intervalle $(-\Delta, \Delta)$.

Soit $W(x) = \int_0^x w(y)dy$, les propriétés de la fonction w énoncées plus haut assurent que $W(0) = 0$ et $W(-x) = -W(x)$. Pour une bulle de taille Δ, on peut donc réduire l'équation 1.12

$$U(x) = W(x + \Delta) - W(x - \Delta) + h. \tag{1.13}$$

Vu que $U(\Delta) = \theta$, on obtient la condition nécessaire et suffisante pour l'existence d'une bulle d'activité :

$$W(2\Delta) + h = \theta \tag{1.14}$$

On peut prouver que la solution est stable si $W'(2\Delta) < 0$ [Amari, 1977, Bressloff, 2012, Pinto and Ermentrout, 2001].

Bulle d'activité induite par un stimulus Les bulles d'activité induites par un stimulus ont été introduites dans le cadre de la modélisation de l'encodage de l'angle de la tête chez le rat [Taube and Bassett, 2003]. Un modèle à été introduit par [Zhang, 1996] basé sur une dynamique d'attracteur très similaire à l'équation d'Amari.

[Xie and Giese, 2002] ont ensuite étudié la dynamique d'un champ neuronal verrouillé sur un stimulus et montré qu'il était possible de trouver une solution analytique dans le cas d'une fonction d'activation Heaviside et qu'il y avait stabilité de la solution pour un certain intervalle de vitesse. Cet intervalle peut être modifié en utilisant un noyau asymétrique qui aura pour effet de donner une préférence directionnelle au DNF.

Au delà de cette limite de stabilité un autre régime apparaît où l'activité du champ tente constamment de rattraper son retard sur le stimulus. Ce régime a été décrit physiologiquement dans des coupes de cerveau sous le terme d'impulsions titubantes (lurching pulses).

Nous avons vu dans cette section comment une analyse mathématique des équations permettait de montrer la dynamique riche des DNF et comment cette dynamique permettait de modéliser des résultats expérimentaux. Nous allons maintenant utiliser la simulation pour montrer les caractéristiques comportementales intéressantes des DNF.

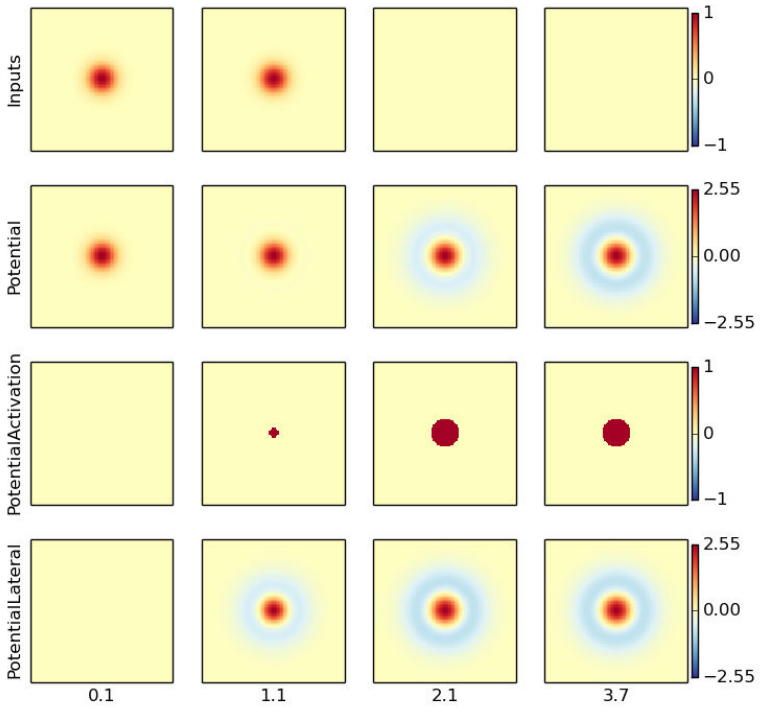


FIGURE 1.9 – Exemple de bulle persistante. Un stimulus de deux secondes (haut) provoque une bulle d'activité (3ème ligne) entretenue par les interactions synaptiques (dernière ligne).

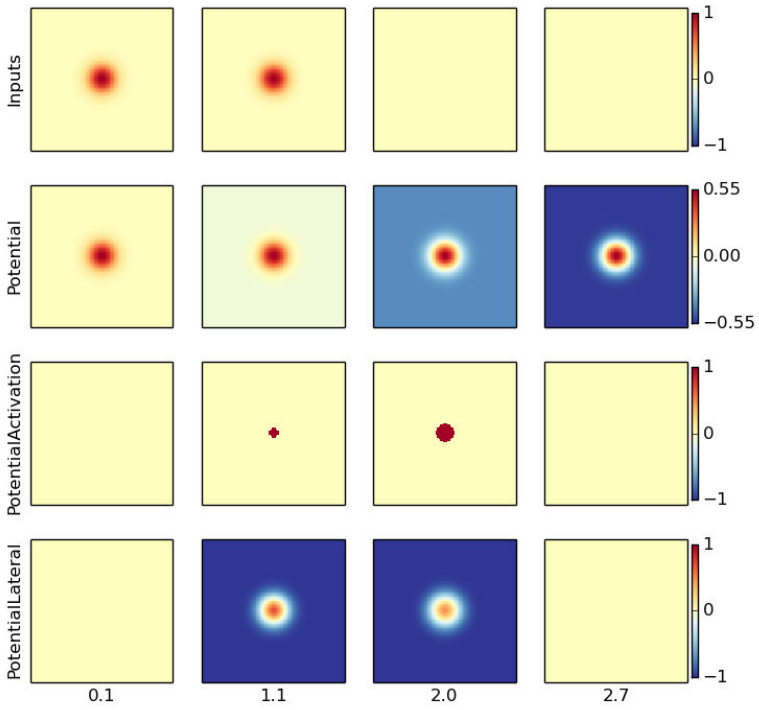


FIGURE 1.10 – Exemple de bulle engendrée par un stimulus. Ici le stimulus de deux secondes ne permet pas un entretiens de l'activité car les interactions latérales ne sont pas assez fortes. L'existence de la bulle d'activité (3ème ligne) dépend donc du stimulus d'entrée.

1.2.1.2 Simulations

Certains comportements dignes d'intérêt sont mieux capturés par la simulation. De plus, étant donné que les modifications que l'on apportera au modèle seront mathématiquement difficiles à analyser nous nous concentrerons par la suite sur les résultats de simulation. Nous allons donc mettre en place un cadre expérimental permettant de quantifier ces caractéristiques lors de différents scénarios basés sur les mécanismes d'attention visuelle.

1.2.1.2.1 Mécanismes de l'attention visuelle «L'attention est un phénomène à la mécanique encore mal connue qui permet de concentrer les ressources de calcul du cerveau sur un objet plutôt qu'un autre. En termes plus philosophiques, c'est la tension de l'esprit vers un objet à l'exclusion de tout autre.» [2]

C'est un mécanisme qui peut être divisé en deux classes. L'attention ascendante (bottom-up) est une attention passive inconsciente issue de notre système cognitif primitif et qui permet de traiter les signaux prioritaires [Posner and Petersen, 1990]. Mais une autre attention, qui elle est volontaire, permet d'accélérer la détection de stimulus par un mécanisme descendant (top-down).

L'étude de l'attention a montré que le mécanisme de base résultait d'une compétition entre les différentes représentations des stimuli dans le cortex temporal inférieur (IT) [Desimone, 1998]. Cette compétition peut être biaisée par des structures différentes dans un cadre descendant. Par exemple il a été montré que le cortex frontal (FEF) inhibait les distractions lorsque l'animal prépare sa saccade oculaire [Desimone, 1998]. Mais nous reviendrons sur la modélisation de ces compétitions biaisées dans la section suivante.

Nous allons ici montrer qu'il est possible de modéliser l'attention visuelle ascendante avec le modèle de champs de neurones d'Amari. Comme dans [Rougier and Vitay, 2006] nous modéliserons le champ visuel comme une carte d'activité bornée entre 0 et 1 avec des stimuli, du bruit et des distractions.

Champs visuels simplifiés Nous allons définir ici les entrées artificielles utilisées tout au long du manuscrit.

Nous appellerons stimulus une bulle gaussienne d'équation :

$$S(x,c) = ke^{\frac{-||x-c||)}{w^2}} \tag{1.15}$$

x est un point de l'espace d'entrée $x \in [0,1]^d$ dans le cas d'une entrée à d dimensions et c est le centre de la gaussienne ($c \in [0,1]^d$). Les paramètres réels k et w sont l'intensité et la taille du stimulus et $||x||$ est la norme euclidienne du vecteur x.

— Bruit. On rajoute un bruit blanc (gaussien) à l'entrée $N(x,z,\sigma)$ dont la probabilité de distribution ne dépend pas de x, i.e.

$$p(z) = \frac{1}{\sqrt{2\pi\sigma_n^2}} e^{-\frac{(z-\mu_n)^2}{2\sigma_n^2}} \tag{1.16}$$

avec μ_n, la moyenne et σ_n l'écart type de la distribution.

— Distractions. On peut ajouter des distractions à l'entrée visuelle. Les distractions sont des stimuli de la même forme que les stimuli principaux mais sont positionnés aléatoirement de façon uniforme dans l'espace d'entrée avec la variable $\kappa \sim U([0,1])^d$

$$D(x,n_d) = \sum_1^{n_d} S(x,\kappa) \tag{1.17}$$

2. http ://atilf.atilf.fr/tlf.htm

TABLE 1.1 – Paramètres des entrées.

Paramètre	Valeur	Description
i_s	1	intensité des stimuli
w_s	0.1	taille des stimuli
μ_n	0	moyenne du bruit gaussien
σ_n	0.01	écart type du bruit gaussien
n_d	0	nombre de distractions

où n_d est le nombre de distractions.

Sauf précision contraire les valeurs des paramètres utilisés sont décrit dans la table 1.1.

Simulation On définit donc l'entrée avec

$$I(x) = \begin{cases} 1 & \text{si } S(x) + N(x) + D(x) > 1, \\ -1 & \text{si } S(x) + N(x) + D(x) < -1, \\ S(x) + N(x) + D(x) & \text{sinon.} \end{cases} \tag{1.18}$$

Il est alors possible de simuler la dynamique du champ neuronal avec cette entrée en réalisant d'abord une discrétisation spatiale et temporelle.

La discrétisation spatiale est au centre de notre intérêt. Comme nous l'avons vu dans l'introduction de cette partie, nous cherchons à étudier la dynamique émergente d'unités de calcul distribuées et décentralisées. Ces unités de calculs sont définies par la discrétisation spatiale de l'équation des champs neuronaux dynamiques.

Nous définissons une grille $\mathcal{G}$. L'équation du potentiel $u(x,t)$ de la cellule située au point x sera :

$$\tau \frac{\partial u(x,t)}{\partial t} = -u(x,t) + \sum_{x' \in \mathcal{G}} w(||x - x'||) F[u(x',t)] + h + I(x,t) \tag{1.19}$$

L'équation des poids latéraux est la suivante :

$$w(d) = k_e e^{\frac{-d}{w_e^2}} - k_i e^{\frac{-d}{w_i^2}} \tag{1.20}$$

Et la fonction d'activation F est la sigmoïde :

$$F(x) = \frac{1}{1 + exp(\beta(u - \theta))} \tag{1.21}$$

où θ est le seuil d'activation et β l'inclinaison de la sigmoïde.

Il est de plus nécessaire de projeter l'espace des entrées et des poids latéraux ($[0,1]^d$) dans l'espace de la grille torique (par défaut) $\mathcal{G}$. Pour cela une interpolation linéaire est effectuée.

La discrétisation temporelle est effectuée à l'aide de la méthode d'Euler. On introduit un pas de discrétisation dt et l'équation calculée à chaque mise à jour des cellules est donc :

$$u_x(t + dt) = u_x(t) + \frac{dt}{\tau}[-u_x(t) + \sum_{x' \in \mathcal{G}} w(||x - x'||) F[u_{x'}(t)] + h + I(x,t)] \tag{1.22}$$

La mise à jour de la grille est effectuée en parallèle. Il faut aussi préciser que la mise à jour des différentes entrées n'est pas nécessairement effectuée en même temps que la mise à jour des

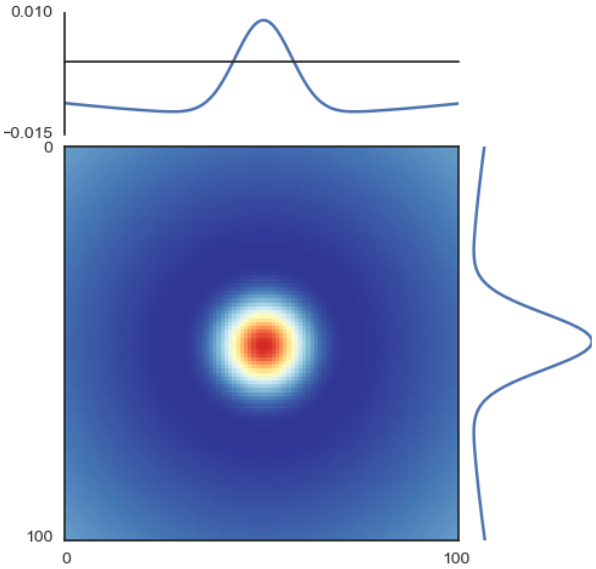


FIGURE 1.11 – Noyau d'interactions latérales utilisé pour les scénarios de compétition.

potentiels mais à une fréquence qui divise la fréquence de mise à jour des potentiels. On appellera la mise à jour des potentiels une *étape de calcul*.

Les paramètres utilisés pour le champ neuronal sont décrits dans la table 1.2 et la différence de gaussiennes correspondante est affichée sur la figure 1.11.

On peut voir la dynamique émergente sur les figures 1.12 et 1.13. La résolution de simulation est $R = 101$. Les autres paramètres sont laissés à leur valeur par défaut.

On voit ici les avantages des champs neuronaux dynamiques pour des applications de traitement d'image. Ils permettent de choisir et de suivre des stimuli de façon émergente mais ils ont aussi l'avantage d'être très robustes aux perturbations. En fait ils fonctionnent comme un filtre spatio-temporel capable de sélectionner un intervalle de fréquence spatiale assez précis déterminé par la forme du noyau des interactions latérales. Ils sont aussi sensibles à des fréquences temporelles précises grâce à leur dynamique temporelle déterminée par τ. En ce

TABLE 1.2 – Paramètres du modèle discret de l'équation d'Amari

Paramètre	Valeur	Description
τ	0.64	constante temporelle du potentiel membranaire
θ	0.75	seuil d'activation des neurones
β	8	inclinaison de la sigmoïde
h	0	potentiel de repos des neurones
k_e	1.25	intensité des poids latéraux excitateurs
k_i	0.7	intensité des poids latéraux inhibiteurs
w_e	0.1	largeur des poids latéraux excitateurs
w_i	1	largeur des poids latéraux inhibiteurs
dt	0.1	discrétisation temporelle de la mise à jour du potentiel

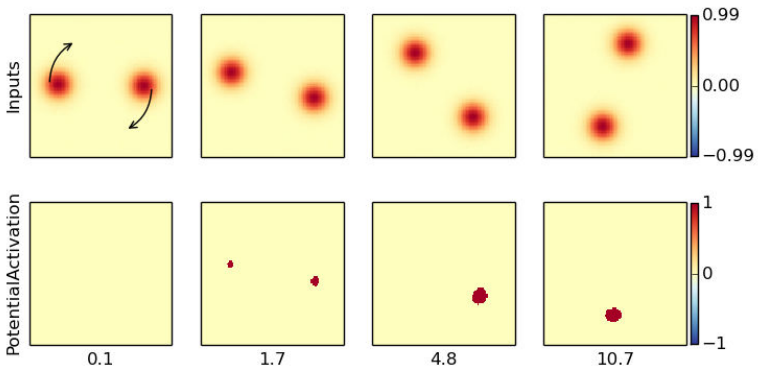


FIGURE 1.12 – Suivi de cible. Deux stimuli tournent à la même vitesse autour du centre de la carte neuronale. Au bout de 2 secondes, l'activation de la carte se déséquilibre et une seule bulle d'activité «gagne» la compétition. Cette bulle va alors suivre son stimulus.

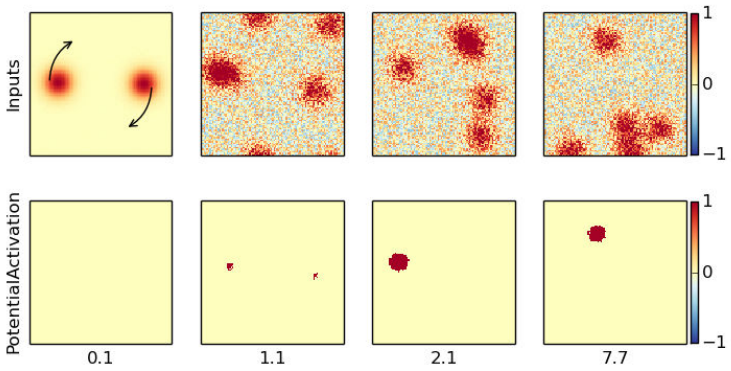


FIGURE 1.13 – Suivi de cible robuste. La dynamique de suivi de cible émerge malgré le bruit et les distractions.

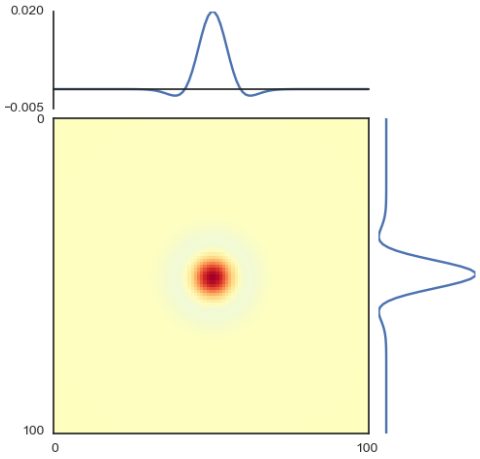


FIGURE 1.14 – Noyau d'interactions latérales utilisé pour le scénario de mémoire de travail

TABLE 1.3 – Paramètres pour la simulation de la mémoire de travail

Paramètre	Valeur	Description
h	-0.02	potentiel de repos
τ	0.08	seuil d'activation
i_e	2.80	intensité des poids latéraux excitateurs
i_i	1.51	intensité des poids latéraux inhibiteurs
w_e	0.07	largeur des poids latéraux excitateurs
w_i	0.09	largeur des poids latéraux inhibiteurs

sens ils sont similaires aux filtres de Gabor spatio-temporels largement utilisés en filtrage vidéo [Adelson and Bergen, 1985], mais avec des comportements globaux émergents supplémentaires, comme la compétition.

Mémoire de travail Il est aussi possible de simuler une mémoire de travail en utilisant les paramètres pour obtenir une bulle d'activité provoquée par un stimulus (voir la section 1.2.1.1.2).

Dans ce cas il est pratique de diminuer la taille de l'inhibition latérale afin de pouvoir retenir plusieurs stimuli dans la mémoire de travail. On peut par exemple utiliser les paramètres décrits dans la table 1.3 dont le noyau correspondant est sur la figure 1.14.

On voit donc que ce modèle simple est capable de simuler une attention visuelle ascendante, et que cette attention est capable de suivre le stimulus sélectionné de façon très robuste et ce malgré le bruit ou les distractions. La mémoire de travail est aussi simplement réalisable en utilisant des résultats analytiques. Il est cependant nécessaire de quantifier cette dynamique afin de pouvoir comparer les performances des différentes versions de l'équation et des implémentations matérielles comme nous le verrons dans le prochain chapitre.

Paramètre	Valeur	Description
c	0	centre de la trajectoire circulaire des stimuli
r	0.3	rayon de la trajectoire circulaire des stimuli
T	36	période de la trajectoire circulaire des stimuli

TABLE 1.4 – Paramètres pour le scénario de suivi de cible

1.2.1.3 Mesure de la dynamique

Pour mesurer la dynamique nous avons mis en place des scénarios d'exécution et des mesures quantitatives associées.

1.2.1.3.1 Scénarios Les scénarios mis en place reprennent les principales observations simulées qui correspondent aux dynamiques utilisées en robotique autonome comme nous le verrons dans la prochaine section. Ces scénarios sont inspirés de [Rougier and Vitay, 2006] et [Quinton, 2010].

Suivi de cible Deux cibles S_1 et S_2 ont une trajectoire circulaire d'équation :

$$\begin{aligned} x(t) =& c + rcos(2\pi\frac{t}{T} + \phi_i) \\ y(t) =& c + rcos(2\pi\frac{t}{T} + \phi_i + 0,25) \end{aligned} \tag{1.23}$$

c est le centre, r le rayon, T la période et ϕ la phase. La phase ϕ_i sera $i0.25$, $i \in \{0, 1\}$.

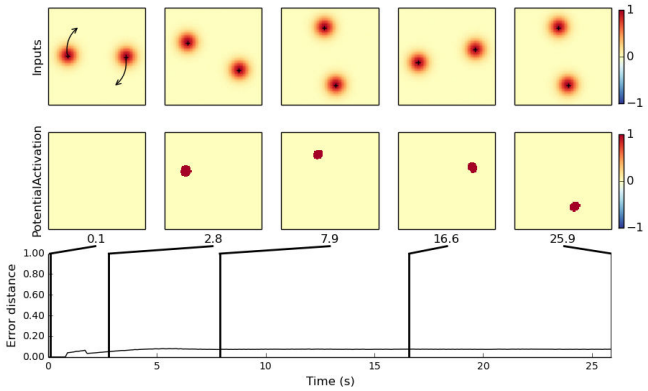
Robustesse Les deux mêmes cibles sont utilisées mais au temps $t = 1s$ on ajoute du bruit à l'image. Soit un bruit gaussien d'écart type 1.0 par défaut, soit un nombre donné de distractions (4 par défaut), soit les deux. Dans tous les cas la mise à jour des distractions se fait toutes les secondes, c'est à dire que les distractions sont statiques pendant 10 pas de calcul des neurones. Le bruit est mis à jour à chaque pas de temps.

Mémoire de travail avec déplacement Le scénario tente de reconstruire les conditions dans lesquelles une stimulation éphémère permet le déclenchement d'une bulle de mémoire entretenue par un stimulus faible mais non nul. Une contrainte supplémentaire est que la mémoire doit suivre le stimulus même quand celui-ci est faible. Ces conditions correspondent au comportement attendu de la carte de mémoire de travail dans le modèle d'attention visuelle simplifié que nous étudierons dans la prochaine section. La vitesse de déplacement est exprimée en unité par seconde ($u.s^{-1}$) car nous n'avons pas défini l'unité de notre espace $[0, 1]$ de taille 1.

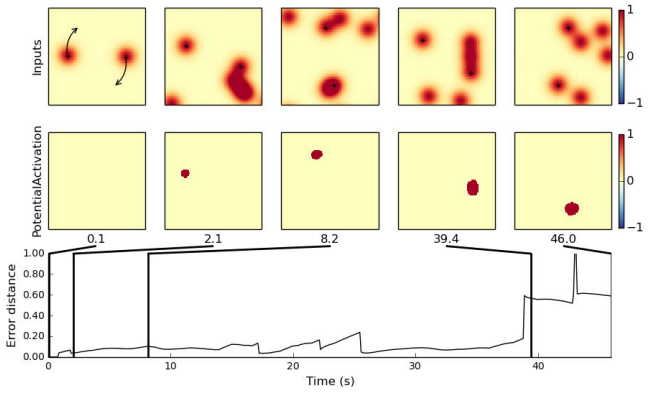
Les étapes que nous avons utilisées sont décrites dans le tableau 1.5 et représentées sur la figure 1.15.

1.2.1.3.2 Caractéristiques mesurées

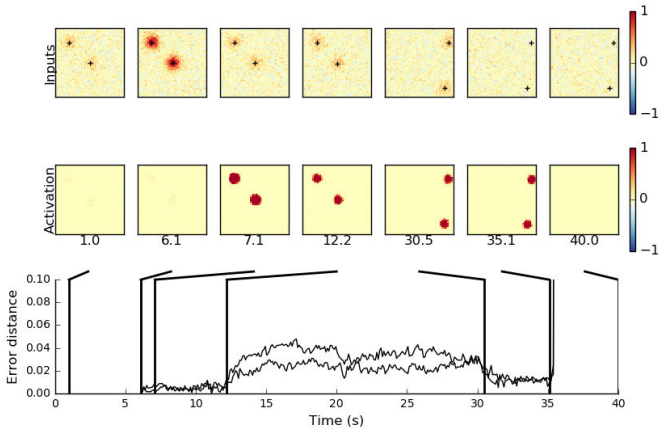
Erreur de distance La principale caractéristique que l'on va étudier sera l'erreur de distance. Cette erreur est définie comme la distance Euclidienne entre le centre de la ou des bulles



(a) Suivi de cible



(b) Robustesse



(c) Mémoire de travail

FIGURE 1.15 – Erreur de distance en fonction du temps. Dans le cas d'un suivi sans distraction, l'erreur reste faible $E < 0.1$. Si il y a des distractions (4 dans ce cas), l'erreur est moins stable et un changement de cible peut arriver (à $t = 39$ ici). Le scénario mémoire de travail montre bien que l'activité commence lorsqu'il y a une augmentation brève des stimuli ($t = 6$) qui permet l'émergence de bulles d'activité capables de suivre les stimuli jusqu'à ce qu'ils disparaissent complètement ($t = 35$).

TABLE 1.5 – Étapes du scénario de mémoire de travail avec déplacement.

Temps	Événement	Description
0.0	initialisation	les stimuli ont une intensité $iLow = 0.2$
6.0	stimulation éphémère	l'intensité des stimuli passe à $iHigh = 1.0$
7.0	fin de la stimulation éphémère	l'intensité des stimuli passe à $iLow = 0.2$
12.0	déplacement	le premier stimulus ce déplace à la vitesse de $trackSpeed = 0.04u.s^{-1}$ verticalement et le deuxième à $trackSpeed/2$ en diagonale
30.0	fin du déplacement	les deux cibles s'arrêtent
35.0	disparition des cible	l'intensité des cibles passe à 0
40.0	fin	fin du scénario

B d'activité du champ neuronal et le centre du ou des stimuli S qui sont sensés être suivis selon le scénario envisagé.

$$E = ||B - S|| \tag{1.24}$$

Les centres des bulles d'activité sont donnés par une méthode de classification DBSCAN (Density-Based Spatial Clustering of Applications with Noise), algorithme de partitionnement des données introduit dans [Ester et al., 1996]. Le barycentre de chaque cluster est alors comparé au centre des stimuli qui sont connus par le simulateur. Il y a trois cas limites à considérer. (1) S'il n'y a pas de bulle cible l'erreur n'est pas comptabilisée ; (2) s'il y a plus de bulles d'activité que de cibles l'erreur est maximale $E = 1$. (3) de même, s'il y a moins de bulles d'activité que de cibles l'erreur est maximale $E = 1$.

1.2.1.3.3 Principaux résultats Nous allons ici rapidement décrire les résultats de l'équation d'Amari dans le cadre expérimental que nous avons mis en place. Ces tests serviront de témoins pour le reste du manuscrit. Les performances des nouveaux modèles développés seront comparés à ces résultats en utilisant le même protocole.

Les modèles sont d'abord optimisés pour la compétition et la mémorisation séparément. L'optimisation pour la compétition est faite avec deux scénarios : suivi et robustesse. L'optimisation pour la mémoire de travail est faite avec le scénario «mémoire de travail avec déplacement» uniquement.

L'algorithme utilisé pour l'optimisation est l'optimisation par essaim particulaire standard avec une topologie adaptative aléatoire à 3 voisins en moyenne [Clerc, 2012]. Les spécificités de l'optimisation sont détaillées en annexe A. En fonction du nombre de paramètres à optimiser la population est initialisée à 20, 50 ou 100 individus. Et l'optimisation se fait sur 100 époques. Une époque met à jour l'ensemble des individus de la population. Les paramètres proposés sont conservés pour l'inertie $w = 0.721$ et pour la cohésion $c = 1.193$.

Le fitness utilisé est calculé à partir de l'erreur moyenne $\bar{E} =< E >$ et de la moyenne du nombre de neurones actifs en dehors d'un cluster cible $\bar{A}$. Afin d'apporter une importance plus forte à l'erreur de distance, la fonction fitness est évaluée par $F = 10\bar{E} + \bar{A}$. Après optimisation chaque condition expérimentale est répétée 50 fois. La moyenne de l'erreur de distance moyenne est alors tracée avec l'intervalle de confiance bootstrap à 95%.

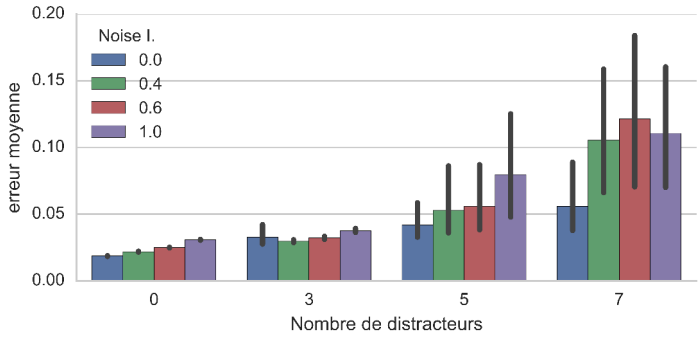


FIGURE 1.16 – Erreur de distance pour le suivi robuste de cible avec différents niveaux de bruit et de distractions.

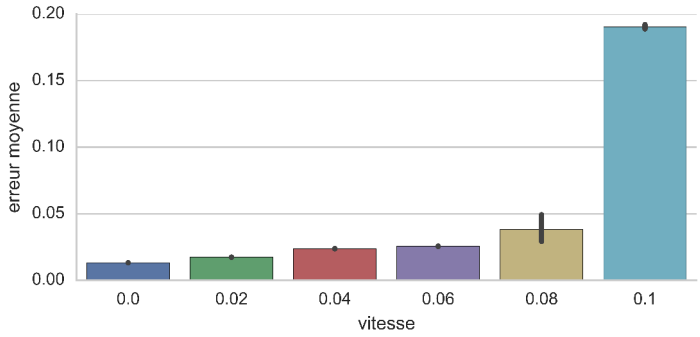


FIGURE 1.17 – Erreur de distance en fonction de la vitesse des cibles suivies par la mémoire de travail. La vitesse est exprimée en unité.s^{-1}.

Suivi de cible et robustesse Pour ces scénarios les paramètres sont optimisés simultanéments sur les deux scénarios pendant 30 secondes. Sur la figure 1.16 on peut voir la tolérance du modèle au bruit et aux distractions.

Mémoire de travail Les paramètres sont optimisés sur le scénario de mémoire de travail avec suivi pendant 40 secondes. On peut observer l'erreur de distance en fonction de la vitesse des cibles sur la figure 1.17 et la robustesse de la mémoire de travail au bruit et aux distractions sur la figure 1.18. Il est intéressant de noter ici la sensibilité de la mémoire de travail aux distractions. Les paramètres ont été optimisés avec un bruit d'intensité 0.1 et par conséquent tous les autres niveaux de bruit donnent des résultats moins bons.

Plus généralement on verra par la suite que le scénario de la mémoire de travail est beaucoup plus sensible et plus difficile à optimiser que les autres.

Nous avons vu de façon analytique ou par simulation trois grands types de comportements émergents. Les fronts progressifs, les oscillations et les bulles d'activité. Nous avons précisé la

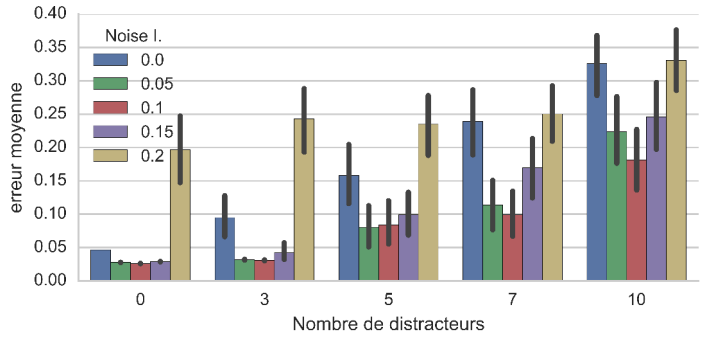


FIGURE 1.18 – Erreur de distance en fonction du nombre de distractions et de l'intensité du bruit.

dynamique des bulles d'activité avec des simulations car elle est d'importance cruciale pour l'encodage des stimuli. Nous avons alors remarqué trois classes de comportement possible : la sélection, le suivi et la mémorisation. Nous avons en outre proposé un cadre de simulation permettant de mesurer quantitativement ces comportements particuliers.

Nous allons maintenant voir comment il est possible de construire des architecture plus complexes en utilisant les cartes de champs neuronaux dynamiques comme blocs de construction.

1.2.2 Cognition et robotique multi-cartes

Les DNF ont été développés pour modéliser la dynamique de l'activation dans le cortex visuel. Mais ce cadre a été étendu aux saccades oculaires [Kopecz and Schöner, 1995], à la planification motrice [Erlhagen and Schöner, 2002], la dynamique d'activation du cortex prémoteur [Bastian et al., 2003] et à la robotique autonome [Bicho et al., 2010, Vitay et al., 2005, Schöner et al., 1995]

Tous ces résultats montrent la versatilité des modèles basés sur la théorie des champs neuronaux dynamiques. Nous allons ici présenter quelques résultats montrant comment des architectures cognitives peuvent être construites pour reproduire des résultats expérimentaux. Mais nous nous intéresserons surtout à l'exploitation de ces concepts en robotique autonome. En effet l'intérêt de notre thèse n'est pas de produire des modèles fidèles à la dynamique corticale mais des modèles capables de calculs performants pour des applications embarquées.

1.2.2.1 Modèles cognitifs

Nous allons voir ici deux modèles cognitifs basés sur les DNF qui sont capables de reproduire des données expérimentales.

1.2.2.1.1 Développement de la mémoire de travail chez l'enfant Piaget a étudié le développement cognitif et en jouant avec ses propres enfants il a découvert que le concept d'objet n'émergeait pas avant 7 ou 8 mois. Avant 7 mois un enfant refuse de chercher un objet donné si on le cache. L'objet a conceptuellement disparu. Entre 7 et 12 mois en revanche l'enfant cherche l'objet, mais pas forcément au bon endroit si l'on bouge spatialement la cachette [Piaget, 2013]. L'expérience fondamentale qui a mis en évidence l'erreur A et non B est la suivante : si l'on cache l'objet dans un petit puits A l'enfant va trouver l'objet dans le puits. On recommence plusieurs

fois et l'enfant trouve l'objet à chaque répétition. On cache alors l'objet dans le puits B situé à une courte distance du puits A, l'enfant va, de façon reproductible, se tromper de puits et chercher l'objet dans le puits A et non le puits B.

Ce qui est surprenant c'est que cette expérience est sensible à de nombreux facteurs (distance entre A et B, lieu de l'expérimentation, type d'objet, etc.). Des modèles à base de DNF on été proposés pour expliquer ces facteurs [Thelen et al., 2001, Simmering et al., 2008].

Le modèle décrit dans [Simmering et al., 2008] par exemple utilise cinq cartes neuronales en interaction et la largeur du noyau d'interaction latérale détermine la distance à laquelle l'enfant fait l'erreur.

1.2.2.1.2 Représentation spatiale de la direction de la tête Dans [Zhang, 1996] Zhang fait l'hypothèse d'un modèle pour les cellules de la direction de la tête chez le rat. Ces neurones sont actifs pour une certaine orientation de la tête dans un repère absolu. C'est à dire que l'activité des neurones est constamment mise à jour lors de la rotation de la tête et du corps de l'animal. Et ce, même dans le noir où l'animal n'est pas capable de se servir de point de repère. Lorsque le rat est désorienté ces cellules ont une activité déviée mais qui peut être recalibré par la vision de points de repère connus de l'animal.

Le modèle proposé par Zhang est un champ neuronal continu sur une seule dimension circulaire. Un attracteur donnera l'orientation qui peut être recalibrée en changeant la symétrie des poids synaptiques.

On a vu dans ces exemples l'intérêt que pouvaient avoir les champs neuronaux dynamiques dans la modélisation de processus cognitifs observés chez les mammifères. Cependant ces modèles sont des hypothèses difficilement vérifiables car même si la dynamique de la population est bio-inspirée, l'architecture des différentes cartes d'activité, leurs paramètres et leurs interconnexions ont peu de support expérimental. Ce sont donc des théories très difficiles à prouver ou a falsifier à cause de l'ampleur des moyens expérimentaux à mettre en place.

En revanche, c'est une inspiration pour les roboticiens puisque ces théories montrent qu'il est possible de reproduire des comportements cognitifs d'assez haut niveau avec une architecture relativement simple et modulaire. C'est plutôt ce type d'application qui nous intéresse dans le cadre de cette thèse.

1.2.2.2 Architecture de systèmes autonomes

La construction de systèmes autonomes en s'inspirant de la cognition animale est l'intérêt premier du domaine de la bio-cybernétique.

Dans ce cadre, les champs neuronaux dynamiques sont un modèle de choix puisqu'ils sont relativement peu coûteux en termes de calcul, ils ont des propriétés suffisamment riches, ils sont bio-inspirés et capables de reproduire des résultats expérimentaux.

Nous allons présenter ici deux exemples d'applications. L'une se base sur une inspiration cognitive pour l'exploration de scène et l'autre introduit un cadre pour la robotique dynamique comportementale.

1.2.2.2.1 Un modèle distribué d'attention visuelle spatiale Dans [Vitay et al., 2005] les DNF sont utilisés afin de simuler une tâche d'attention visuelle relativement complexe : l'exploration visuelle de scène. Il y a plusieurs types d'exploration possibles. La première est dite

parallèle et permet de discriminer dans un temps très court les objets afin de trouver l'objet cherché. Ceci est possible si cet objet a des caractéristiques suffisamment uniques dans la scène pour être reconnu directement.

L'autre type de recherche est la recherche séquentielle. Elle oblige l'agent à évaluer tous les objets ressemblant à la cible un par un. Cette recherche séquentielle met en jeu plusieurs mécanismes importants du système cognitif. Le premier est la sélection. Il est nécessaire de choisir où commencer la recherche, et où la continuer ensuite. Cela veut dire qu'il est nécessaire de pouvoir sélectionner un objet parmi des objets similaires. Le deuxième est la gestion des saccades oculaires. Elle doivent être contrôlées précisément afin de concentrer la fovea sur chaque objet le plus rapidement possible. Le troisième est la mémoire de travail. Une exploration de scène séquentielle suppose en effet d'être capable de retenir les objets déjà envisagés afin de ne pas les choisir comme prochaine cible.

On voit donc que cette application est très riche cognitivement et qu'elle parait tout à fait adaptée à l'utilisation des DNF vu qu'ils ont toutes les caractéristiques requises.

En pratique il faut donc utiliser une carte de mémoire de travail qui va retenir la position des objets déjà observés et inhiber les bulles correspondantes dans la carte de focalisation de l'attention (carte Focus). La carte Focus va choisir une cible et amorcer une saccade oculaire vers cette cible. La cible sera ajoutée à la mémoire de travail et le choix d'une nouvelle cible peut recommencer jusqu'à ce qu'il n'y ait plus de cible disponible dans la carte Focus. Le déclenchement d'une nouvelle saccade est contrôle par les cartes modélisant le striatum et une carte de récompense.

L'avantage de l'utilisation des DNF dans ce cadre est qu'il est possible de bouger les objets pendant la recherche et la dynamique de « suivi de cible » des DNF permettra de mettre à jour leur position aussi bien dans la carte Focus que dans la carte de mémoire de travail.

1.2.2.2.2 Version simplifiée Dans le cadre de cette thèse j'ai étudié une version simplifiée du modèle décrit ci-dessus. Cette version simplifiée se concentre sur la mémoire de travail et le focus à la base de la mémorisation des stimuli lors de l'exploration séquentielle de la scène.

La carte Focus est connectée au stimulus et à la mémoire de travail. La mémoire de travail est activée lorsque la carte Focus est activée et un stimulus est présent. Cette activation construit une bulle mémorisant la position du stimulus même si l'activation de la carte Focus disparaît. Cette mémoire permet l'inhibition en retour de la carte Focus afin qu'elle dirige son attention vers un autre stimulus (figure 1.19).

Outre le mécanisme des saccades oculaires qui n'est pas implémenté ici, la différence fondamentale entre cette architecture et celle décrite plus haut est le mécanisme de la mémoire de travail. Dans leur modèle, [Vitay et al., 2005] considéraient deux DNF en miroir maintenant l'activité des bulles par auto-excitation. Dans l'architecture proposée ici un seul DNF fait office de mémoire de travail comme dans [Fix et al., 2007] et il n'y pas de mécanisme pour attendre un événement avant de porter l'attention sur un autre objet. Cette architecture parcourt la scène objet par objet le plus rapidement possible.

Cette architecture permet de tester la dynamique des DNF dans un contexte plus complexe et donc plus difficile à optimiser que dans les scénarios plus élémentaires décrits en 1.2.1.3.

1.2.2.2.3 La dynamique du comportement L'avantage de l'approche de DNF pour la robotique autonome est qu'elle est basée sur des approches de robotique comportementale préexistantes et qu'elle se base sur les propriétés des systèmes dynamiques pour construire les différents comportements [Schöner et al., 1995].

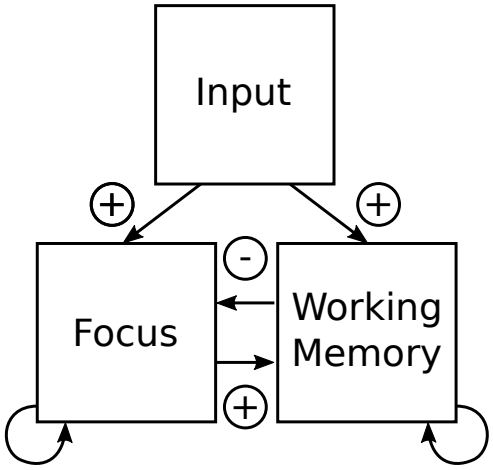


FIGURE 1.19 – Architecture à deux cartes DNF pour une attention sélective. La carte Focus porte son attention sur un seul stimulus alors que la carte Working Memory sauvegarde la position de tous les stimuli déjà considérés.

L'objectif n'est pas ici de donner un tutoriel complet de la théorie des champs dynamiques mais d'expliquer comment les DNF peuvent être utilisés pour le contrôle des comportements d'un agent autonome.

Principe général Chaque DNF représente un comportement (ou fonction) défini sur un espace de comportements possibles. On dit que le comportement est instancié quand il y a une ou plusieurs bulles d'activité. Ces différents comportements sont connectés par l'intermédiaire d'excitations ou d'inhibitions [Engels and Schöner, 1995].

On peut utiliser différents régimes de DNF (cf dynamique des DNF). Le régime transitoire : la bulle d'activité ne peut être maintenue que lorsqu'il y a stimulation. Le régime stable : si une bulle d'activité apparaît elle sera maintenue (on peut donc mémoriser la position du stimulus).

On peut aussi utiliser différents types d'interactions latérales.

— Inhibition globale : il ne peut y avoir qu'une seule bulle d'activation.
— Inhibition locale : il peut y avoir plusieurs bulles.

Par exemple une question intéressante est la distance à laquelle deux bulles peuvent co-exister dans le cas d'une inhibition locale . Si deux stimuli sont séparés d'une distance plus petite que d_{max}, leurs bulles d'activité fusionnent. On peut représenter ce phénomène avec l'évolution du centre de chaque bulle d'activité en fonction du temps lorsque les stimuli fusionnent (fig .1.20).

L'amplitude des stimuli peut moduler cette instabilité car si l'un des stimuli à une amplitude beaucoup plus élevée que l'autre, la fusion sera asymétrique. Cela peut par exemple permettre à un agent de prendre une décision lorsque deux obstacles sont proches. Si la distance est inférieure à une distance seuil alors les deux bulles encodant la présence des obstacles sont fusionnées. L'agent décidera donc de contourner les deux obstacles car l'espace entre les obstacles est trop petit pour l'agent. Cela nécessite évidement de moduler les paramètres des DNF en fonction de la dynamique souhaitée. Dans l'exemple de l'intervalle inter-obstacles il faut choisir la largeur des bulles d'activité en fonction de la taille du corps de l'agent.

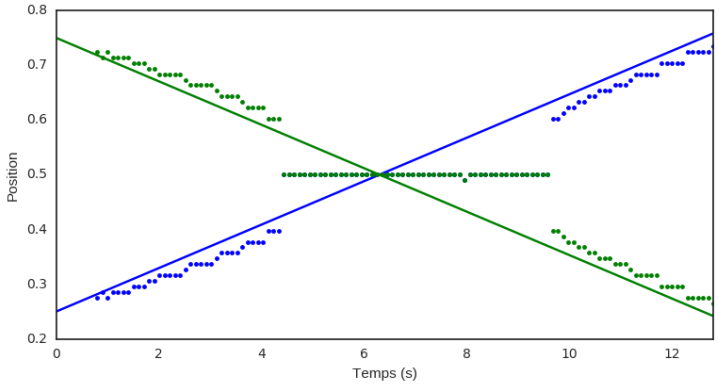


FIGURE 1.20 – Trajectoire de deux cibles (ligne bleue et ligne verte) et du barycentre des bulles d'activité associées (points bleus et points verts). Lorsque les deux bulles sont suffisamment proches, elles fusionnent (ligne verte du milieu).

Navigation autonome Nous allons montrer ici comment une dynamique d'attracteurs et de répulseurs peut être mise en place. Ce n'est pas exactement ce qui est montré dans [Schöner et al., 1995] et [Engels and Schöner, 1995] mais les grands principes sont les mêmes et en sont directement inspirés.

La tâche à laquelle nous allons nous intéresser est la navigation autonome dans un environnement inconnu et dynamique. Le robot aura pour tâche d'éviter les obstacles et de se diriger vers une cible. On utilise un simulateur de robot (comme v-rep par exemple [3]) avec lequel on va contrôler un robot e-puck. La variable de comportement est la direction angulaire de la cible ϕ dans un repère égocentrique.

Nous allons utiliser deux DNF uni-dimensionnels. L'un encode la direction des obstacles F_o et l'autre la direction des cibles F_c. Par extension nous appellerons ψ_o et ψ_c le barycentre de l'activité des deux champs précédemment cités.

Le champ cible contrôle les neurones moteurs à l'aide d'une projection (excitatrice) sur la vitesse de rotation des roues droite et gauche (r_d et r_g resp.) avec

$$r_d(\psi_c) = \frac{2}{(1+\exp(-10\psi_c))} - 1 \tag{1.25}$$

$$r_g(\psi_c) = -r_d(\psi_o) \tag{1.26}$$

comme le montre la figure 1.21.

La vitesse de rotation $\dot{\phi}$ d'un robot deux roues est définie avec

$$\dot{\phi} = \frac{r}{L}(r_d - r_g) \tag{1.27}$$

où r est le rayon des roues et L la distance entre les roues. On a donc l'équation dynamique de la variable ϕ :

$$\dot{\phi} = \frac{r}{L}(2r_d) \tag{1.28}$$

3. http://www.coppeliarobotics.com/

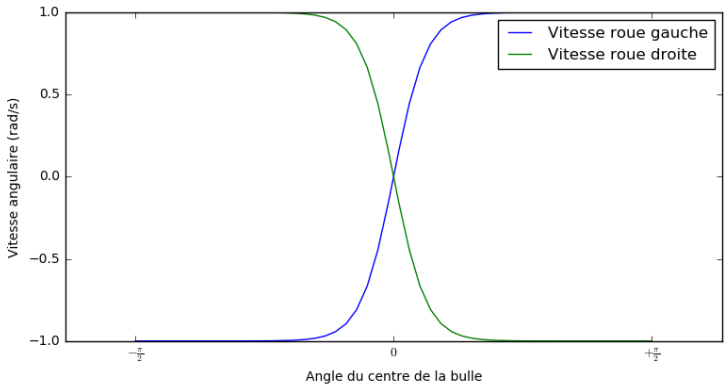


FIGURE 1.21 – Vitesse des roues en fonction de l'angle ψ de la position de la bulle cible sur le DNF.

$$\dot{\phi} = \frac{r}{L}(2r_d) \tag{1.29}$$

On peut donc tracer la relation entre la vitesse de rotation et l'angle de la cible sur la figure 1.22. En faisant l'hypothèse que le barycentre du DNF correspond exactement à l'angle de la cible $\phi = \psi_c$ on a un point de stabilité quand la courbe coupe l'axe des abscisses. La pente est négative, ce qui montre la présence d'un attracteur. En effet si ψ s'éloigne un peu de cet attracteur, la rotation du véhicule va le ramener dans la position stable. En inversant simplement les contributions aux roues droite et gauche, on peut construire un répulseur que l'on utilisera pour le DNF des obstacles.

Cette dynamique est suffisante pour éviter les obstacles et aller vers la cible.

Nous avons vu dans cette section comment il était possible de construire des architectures de niveau relativement élevé afin de mettre en place des dynamiques contrôlant un robot autonome.

Ces approches ne sont pas parfaites vu qu'elles souffrent (comme toutes les approches comportementales) du problème de l'architecture. Le problème de l'architecture est le fait qu'il est très difficile d'organiser l'interaction des comportements de bases pour arriver à des comportements plus haut niveau.

Néanmoins ces architectures sont crédibles et démontrent les pouvoirs applicatifs des champs neuronaux dynamiques. De nombreux travaux ont de plus prouvé que les DNF avaient un pouvoir de décision proche d'un encodeur optimal [Deneve et al., 2001]. Il est donc tout à fait justifié d'utiliser les DNF pour effectuer ce type de calcul.

* * *

Nous avons donc vu les différents types d'applications envisageables avec les DNF. Les propriétés mono-cartes sont déjà intéressantes pour le suivi visuel et possèdent en plus des capacités émergentes de sélection et de mémorisation. De plus, ces simples propriétés sont suffisantes pour

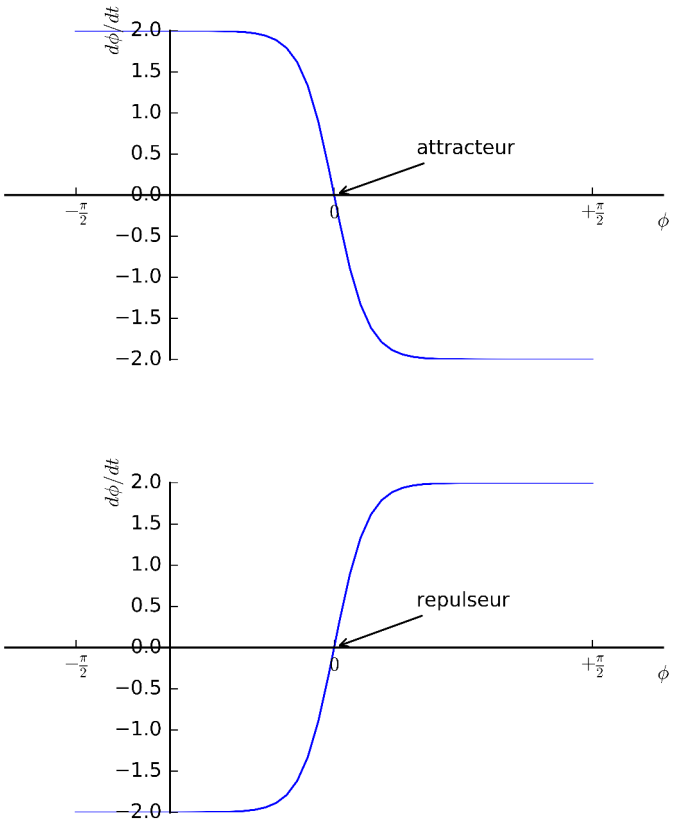


FIGURE 1.22 – Vitesse de rotation en fonction de l'angle de la cible (haut) ou d'un obstacle (bas).

construire des architectures relativement complexes permettant de reproduire des résultats expérimentaux (développement cognitif, préparation motrice, saccades oculaires et cellules d'orientation) ou pour construire des agents autonomes évoluant dans un environnement dynamique.

Dès lors, nous comprenons pourquoi ces modèles sont des bons candidats pour un calcul émergent et robuste. Une dynamique assez simple de compétition et coopération émerge d'interactions entre unités de calcul relativement simples (les cellules de la version discrète du DNF). A partir de cette dynamique, de nombreux calculs sont réalisables de façon complètement parallèle et décentralisée.

Nous allons maintenant voir quelles sont les contraintes liées à l'implémentation matérielle des DNF et comment y remédier.

Chapitre 2

Implémentations matérielles des champs neuronaux dynamiques

Malgré la relative popularité des champs neuronaux dynamiques, il n'existe pas d'implémentation matérielle véritablement dédiée à leur exécution. Cependant il est tout à fait envisageable d'utiliser des architectures matérielles de réseaux de neurones déjà existantes. En effet la version discrète des champs neuronaux dynamiques n'est qu'un type de réseau de neurones récurrent complètement connecté comme les réseaux de Hopfield par exemple. Nous allons donc dans cette partie faire un court résumé des architectures de réseaux de neurones sur FPGA afin d'en extraire les principes conducteurs utiles pour l'implémentation numérique des champs neuronaux dynamiques.

Étant donné qu'il existe un besoin pour des implémentations FPGA de champs neuronaux dynamiques nous en profiterons aussi pour proposer une implémentation inspirée des principes des implémentations neuromimétiques. Ce besoin d'accélération matérielle est assez récent et répond à des contraintes d'intégration (intégration à une architecture déjà développée sur FPGA) et à des contraintes de compacité (pour du calcul embarqué). C'est le cas par exemple de cette architecture matérielle FPGA [Maggiani et al., 2016] où un DNF est utilisé comme filtre final pour améliorer la fiabilité d'une architecture de classification utilisée pour la détection de piétons.

Même si des GPU (graphical processor units) existent pour faire du calcul embarqué (NVIDIA Tegra par exemple) le calcul des champs neuronaux exige une fréquence élevée pour les entrées et les sorties. Le calcul GPU n'est donc pas le plus adapté [Rostro-Gonzalez et al., 2010].

Pour l'instant le besoin pour des architectures à grande échelle impliquant plusieurs dizaines de cartes DNF n'est pas très marqué puisque les architectures pour la robotique ou la vision sont encore au stade de développement. Cependant nous veillerons quand même à étudier la mise à l'échelle de nos architectures dans l'objectif de pouvoir dans le futur facilement intégrer plusieurs cartes.

Nous verrons dans une première section les principaux types d'implémentation pour les réseaux de neurones sur FPGA, nous décrirons ensuite comment l'inspiration corticale permet de définir des principes d'implantation économes. A partir de ces principes nous allons modifier le modèle canonique des DNFs afin d'y intégrer un calcul à base d'impulsions. Cela permettra de simplifier grandement l'implémentation matérielle et permettra de reprendre les principes architecturaux du calcul neuromimétique basé sur des neurones impulsionnels. Nous pourrons ainsi décrire les deux architectures centralisées que nous avons conçues pendant cette thèse, afin de les comparer dans les prochains chapitres aux modèles décentralisés qui en constituent la contribution principale.

Il y a beaucoup de types de substrat matériel possibles pour implémenter un calcul neuronal. Même en envisageant uniquement les technologies «conventionnelles» [4], il faut différencier les architectures numériques des architectures analogiques (à base de CMOS ou autre). Il y a beaucoup de travaux étudiant les propriétés neuromimétiques des substrats analogiques car il est possible de reproduire précisément le comportement de neurones et de synapses grâce à un ensemble réduit de composants sur des cartes d'intégration à large échelle (very large scale implementation, VLSI en anglais). Il est donc possible d'avoir une grande densité de neurones en parallèle tout en ayant une consommation énergétique très compétitive. En revanche ces circuits ont le défaut d'être bruités de façon importante limitant donc les applications possibles [Indiveri et al., 2011] ou nécessitant la conception de nouveaux paradigmes de calculs. Nous allons voir au cours de ce manuscrit que certaines de nos implantations sont très bruitées et que les calculs que nous effectuons sont néanmoins suffisamment robustes et «supportent» un niveau de bruit élevé. Il est donc très probable que les substrats analogiques soient adaptés à nos applications. Cependant ils ne seront pas envisagés dans ce manuscrit qui prend le parti pris de se consacrer aux implémentations numériques.

2.1 Les réseaux de neurones matériels

Nous allons présenter ici les concepts fondamentaux de la conception de réseaux de neurones sur circuit matérielle numérique. Le but n'est pas de faire une revue complète de ce sujet qui est très large. Le lecteur pourra se référer à [Misra and Saha, 2010].

Nous nous concentrerons sur les approches FPGA et sur les approches neuromimétiques de grande échelle.

Nous appellerons réseau de neurones artificiels (ANN pour Artificial Neural Networks) un graphe de neurones connectés les uns avec les autres par l'intermédiaire de synapses. La version la plus simple d'un réseau de neurones est un perceptron (fig. 2.1) mais nous ne nous limiterons pas à cette architecture. Dans le modèle du perceptron les entrées sont intégrées par des neurones via des poids synaptiques. Une fonction de transfert non linéaire produit la sortie du neurone. Il est alors possible d'apprendre les poids synaptiques nécessaires à une certaine fonction à l'aide d'un apprentissage supervisé par exemple. Il existe de nombreuses classes de réseaux de neurones pouvant être instanciées à partir de ce type de modèle : perceptrons multicouches, réseaux de neurones récurrents et réseaux impulsionnels.

Dans ces réseaux, chaque arc du graphe possède un poids synaptique qui doit être implémenté par une multiplication. C'est donc un point critique pour la surface d'implémentation puisque la multiplication est coûteuse en numérique.

2.1.1 Les FPGA comme substrat de calcul

Les circuits FPGA (pour Field Programmable Gate Arrasy) sont des substrats numériques reprogrammables et sont une cible de choix pour faire du prototypage car ils constituent un compromis entre des architectures ASIC très performantes mais au coût élevé et les microprocesseurs peu coûteux mais pas assez flexibles. La description de la configuration matérielle souhaitée utilisée pour programmer le circuit FPGA peut être générée à partir d'un langage de description matérielle tel que Verilog ou VHDL. Un tel langage est d'assez haut niveau (puisqu'il permet de faire des boucles par exemple) et accélère donc le prototypage.

4. on ne parlera pas ici des architectures dites «non-conventionnelles» comme les ordinateurs chimiques, à ADN, optiques ou quantiques.

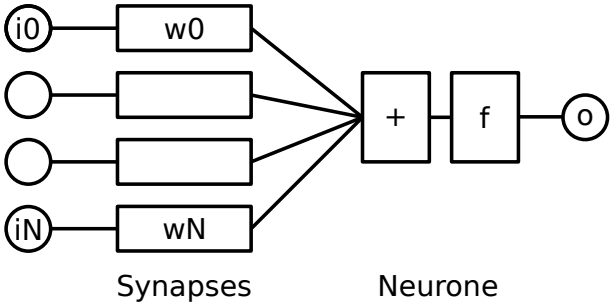


FIGURE 2.1 – Le perceptron comme modèle générique pour les réseaux de neurones. Les entrées $(i_0...i_n)$ sont multipliées par les poids synaptiques $(w_0...w_n)$ avant d'être intégrées dans le neurone et transformées par une fonction de transfert f. La sortie o est ainsi générée et peut être liée à d'autres neurones via des synapses.

Concrètement, un circuit FPGA offre un nombre élevé de ressources de calcul et d'éléments mémoire élémentaires librement et indépendamment configurables, connectées par un réseau également configurable. Typiquement, un FPGA Xilinx de série 6 ou 7, est structuré en une grille de blocs logiques configurables (Configurable Logic Block ou CLB en anglais). Ces CLB contiennent deux "slices" qui sont connectées à un "switch". Chaque slice contient quatre "look-up tables" (LUT) à 6 entrées et huit bascules [Xilinx, 2014].

Les LUT sont programmées à l'initialisation du circuit afin de calculer une fonction logique à 6 entrées. Leur association via le réseau configuré permet de réaliser n'importe quel circuit logique, pour peu que le nombre de ressources soit suffisant. Les bascules permettent de stocker des éléments de logique séquentielle. Les slices contiennent aussi de la RAM distribuée, de la logique optimisée pour les opérations à retenue et parfois des registres à décalage.

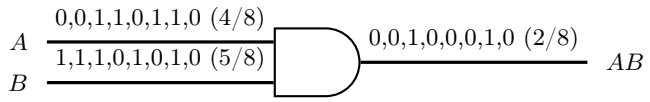
Les circuits FPGA permettent donc d'implémenter quasiment n'importe quelle architecture matérielle sous certaines limites de ressources imposées par la dimension et la performance du circuit.

Le premier avantage d'utiliser des circuits FPGA dans le cadre des réseaux de neurones est de pouvoir exploiter la nature parallèle du calcul neuronal de façon beaucoup plus efficace que ne le permettent des processeurs ordinaires. Les circuits FPGA permettent un grain de calcul très fin tout en ayant des capacités d'interconnexions très denses ce qui est idéal pour des implantations neuronales.

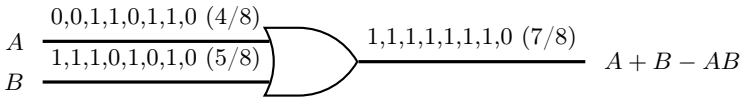
L'autre grand avantage est la compatibilité ascendante d'une implémentation FPGA. A moins d'utiliser des propriétés très caractéristiques d'une carte FPGA, il est très probable qu'une architecture soit synthétisable sur les futurs circuits FPGA sans trop d'effort. On bénéficie ainsi des progrès technologiques avec un coût réduit.

Nous allons souligner les principes majeurs de l'implantation des réseaux de neurones sur ce substrat.

Il existe de nombreux types de parallélisme dans le cadre des réseaux de neurones qui peuvent être exploités lors d'une implémentation matérielle. On peut distinguer le parallélisme de session d'apprentissage, le parallélisme d'exemple d'apprentissage, le parallélisme de couche, le parallélisme de neurone et le parallélisme de connexion [Girau, 1999]. Il est donc possible d'exploiter chacun de ces niveaux afin d'accélérer les calculs lors des phases d'apprentissage ou d'exploitation. Dans ce chapitre nous appellerons implémentation parallèle une implémentation *complètement*



(a) Multiplication



(b) Addition diminuée

FIGURE 2.2 – Exemples d'opérateurs arithmétiques à flux de bits. La multiplication est implémentée par une porte ET et une fonction d'addition diminuée avec une porte OU. La valeur asymptotique décrite comme des opérations sur A et B diffère de la valeur réelle décrite entre parenthèses par une fraction en fonction de la longueur du flux de bits (ici $n = 8$).

parallèle qui profite du parallélisme neuronal le plus fin : les neurones et les synapses. Une application parallèle dédie des ressources et de la logique physique pour chacune de ces unités de calcul et le tout est interconnecté avec des connexions dédiées. Ces architectures utilisent donc en général beaucoup de surface mais sont très rapides.

Au contraire nous appellerons implémentation à multiplexage temporel les implémentations qui ont une partie non parallèle (un neuroprocesseur par exemple). Ces implémentations peuvent intégrer également un certain degré de parallélisme, mais avec un grain de calcul plus grossier.

2.1.2 Implémentation parallèle

Ce type d'implémentation correspond à une exploitation directe de ce qu'on désigne couramment par parallélisme neuronal, au sens où il s'agit d'une parallélisation directement issue de la structure distribuée du graphe qui détermine l'architecture fondamentale du réseau de neurones. Néanmoins, ce parallélisme naturel ne s'adapte pas naturellement à la structure parallèle des circuits supports, principalement en raison d'interconnexions trop denses ne respectant pas la structure bi-dimensionnelle des circuits (malgré les multiples couches métalliques), et en raison de la présence de très nombreux opérateurs coûteux en surface d'implantation. Les prochains chapitres se concentreront sur les solutions qu'on peut apporter aux problèmes d'interconnexion dense dans le cadre des DNF. Dans cette section, on se focalisera sur les solutions permettant de réduire les surfaces d'implantation.

Afin de diminuer la surface des implémentations parallèles il est courant d'éviter d'utiliser des multiplieurs que ce soit pour les synapses ou pour les neurones. Il y a pour cela plusieurs stratégies, par exemple utiliser des puissances de 2 afin de ramener la multiplication à un décalage de bits ou bien changer d'arithmétique (arithmétique sérielle, arithmétique en-ligne ou arithmétique à flux de bits par exemple) [Maguire et al., 2007].

Cette section se concentre sur les approches utilisant des flux de bits mais il existe évidemment beaucoup d'autres solutions. L'arithmétique à flux de bits sera utilisée dans le dernier chapitre de cette thèse. Il est donc utile de détailler ici les enjeux de cette approche.

L'arithmétique à flux de bits (aussi appelée calcul stochastique ou arithmétique à base d'impulsions) a été introduite en 1960 comme une alternative aux calculs conventionnels binaires. L'idée est de représenter les nombres sous forme de probabilités numériques [Gaines, 1967].

Soit un nombre $x \in [0, 1]$. Il sera encodé dans un flux de bits $b_1, ..., b_n$ de taille n en générant n bits successifs avec une probabilité $p_x = x$ pour que le bit soit haut. L'estimation de x, $\hat{x}$ est

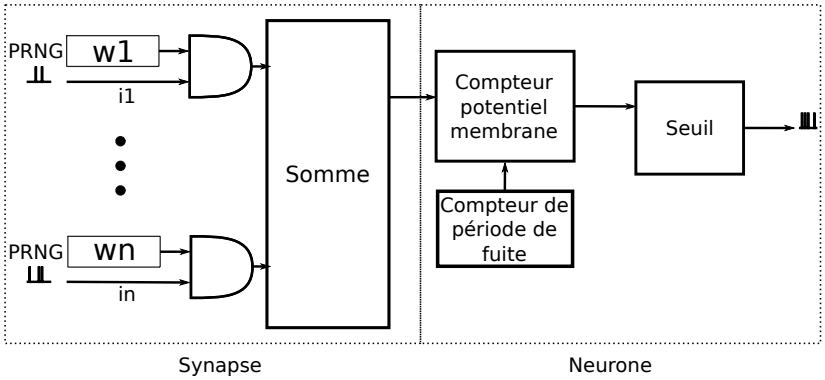


FIGURE 2.3 – Implémentation d'un réseau de neurones impulsionnels à l'aide d'une arithmétique à flux de bits. Aucune multiplication n'est nécessaire.

calculée en «décodant» le flux de bits avec la moyenne des bits :

$$\hat{x} = \frac{1}{n} \sum_{i=1}^{n} b_i \tag{2.1}$$

L'avantage principal est de pouvoir faire les multiplications avec une porte ET comme décrit sur la figure 2.2. La porte OU quand à elle permet d'additionner les flux de bits avec une erreur. Pour encoder une valeur x en flux de bits il faudra donc utiliser un générateur de nombres pseudo-aléatoires (pseudo random number generator, PRNG en anglais) avec un comparateur.

On peut voir sur la figure 2.3 comment une arithmétique à base de flux de bits peut être utilisée pour implémenter un réseau de neurones [Maguire et al., 2007]. Chaque poids synaptique est généré sous forme de flux de bits grâce à un générateur de nombres pseudo-aléatoires et est ainsi multiplié par l'entrée. La logique du neurone se charge ensuite d'intégrer les potentiels synaptiques (par l'intermédiaire d'une somme par exemple) et un seuil sert de fonction d'activation.

Dans [Joubert et al., 2012] l'auteur propose aussi un mécanisme pour la fuite membranaire, dans le cas d'une implantation de neurone impulsionnel. Il propose d'utiliser un compteur qui décrémente le potentiel membranaire avec une période donnée.

Il est comme cela possible d'implémenter des modèles plus ou moins complexes de neurones avec des fonctions d'activation variées comme la tangente hyperbolique [Bade and Hutchings, 1994]. Il existe une grande variété d'encodages et d'opérateurs possibles pour une arithmétique stochastique ouvrant un large spectre d'optimisations possibles [Alaghi, 2015]. Il est intéressant par exemple d'étudier l'influence du PRN utilisé lors de l'encodage. Les opérateurs principaux peuvent être complètement modifiés si les opérandes sont encodées avec des PRN corrélés ou anti-corrélés. Souvent on s'assurera que les flux sont non corrélés mais utilisés à bon escient, les flux corrélés ou anti-corrélés permettent de définir de nouveaux opérateurs [Alaghi, 2015]. Une autre propriété intéressante est la précision progressive. La précision progressive décrit simplement le fait que plus le décodage du flux de bits est long plus la valeur est proche de la valeur moyenne asymptotique. Cela permet une convergence accélérée car même après un temps court de décodage, on a déjà une bonne approximation de la valeur. Cette propriété peut être améliorée par l'utilisation de générateurs de nombres à faible divergence [Dalal et al., 2008] remplaçant les générateur de nombres pseudo-aléatoires.

Le principal défaut de ces architectures est la surface nécessaire à la génération des nombres aléatoires. On souhaite en général s'assurer qu'il n'y a aucune corrélation entre les flux de bits

utilisés dans un opérateur. Il faut donc s'assurer que chaque flux de bits a une source aléatoire différente. Il est souvent en plus nécessaire de briser la corrélation en cours de calcul par des PRNG supplémentaires.

L'autre défaut d'une arithmétique à flux de bits est le nombre de bits nécessaires pour atteindre une précision aussi grande qu'un encodage binaire [Manohar, 2015]. Prenons un exemple. On veut encoder des nombre $x \in [0, 1]$. La précision est correcte si la différence entre $\hat{x}$, la représentation de x et x est inférieure à ϵ :

$$|\hat{x} - x| \leq \epsilon \tag{2.2}$$

Pour un encodage binaire on peut utiliser une représentation à point fixe afin d'obtenir $\epsilon = 2^{-(b+1)}$ où b est le nombre de bits.

L'on souhaite que l'encodage stochastique préserve cette précision avec un certain seuil d'erreur δ. On veut donc que la propriété suivante soit vérifiée :

$$\max_{x \in [0,1]} Pr_{\hat{x}}[|x - \hat{x}| > \epsilon] \leq \delta \tag{2.3}$$

Par exemple, si l'on souhaite que la probabilité d'erreur δ soit inférieure à 0.1 il faut utiliser un flux de bits de longueur $n = 2752$ pour avoir la même précision qu'avec un encodage binaire sur 5 bits [Manohar, 2015].

2.1.3 Multiplexage temporel

Le multiplexage temporel est à la base de beaucoup d'implémentations matérielles car il permet en général un bon compromis surface/vitesse et donc des réseaux de taille plus importante que les approches parallèles à grains fins.

Le principe de base se rapproche d'une architecture matérielle de type von Neumann où les calculs sont séparés de la mémoire. Des machines à états finis où des processeurs dédiés mettent-à-jour des potentiels synaptiques et neuronaux d'une façon séquentielle. Il est alors possible de paralléliser les processeurs afin d'optimiser le débit de calcul souhaité. On a ainsi un degré de parallélisme plus grossier.

L'avantage par rapport à une implémentation complètement parallèle est la précision car la relative centralisation des calculs dans les différents processeurs de l'implémentation permet l'utilisation de précisions plus importantes et l'implémentation de modèles de neurones plus complexes [Maguire et al., 2007].

La précision des calculs est un paramètre essentiel à prendre en compte dans les réseaux de neurones. Dans le cas d'une arithmétique classique, l'ingénieur a le choix entre deux représentations : virgule fixe ou virgule flottante. Étant donné que la précision requise par les réseaux de neurones n'est pas très grande on va souvent préférer l'utilisation de virgule fixe qui est beaucoup moins coûteuse. Un nombre réel sera donc représenté par un certain nombre de bits avant et après la virgule. On peut par exemple noter la précision e, f avec e le nombre de bits encodant la partie entière et f le nombre de bits encodant la partie fractionnaire. Par exemple si on choisit $e = 3$ et $f = 7$, le nombre 1011001101 sera noté $101,1001101$ et aura pour valeur décimale si on choisit un codage non signé $717\frac{1}{2^7} = 5.6015625$. Le nombre de valeurs possibles est 2^{10}, la valeur minimale est $0,0$ et la valeur maximale est $7,9921875$. Il est bien entendu possible d'étendre cette notation aux nombres négatifs, en codage signé ou en complément à 2.

La question de la précision à utiliser pour un réseau de neurones dépend principalement de l'application que l'on veut en faire. Dans le cas des perceptrons multicouches, il a été montré

qu'il y a souvent un compromis entre le taux de convergence et la précision. Avec certaines hypothèses il a été montré que 16 bits étaient suffisants et 12 bits essentiels pour un algorithme de rétro-propagation du gradient. Sans phase d'apprentissage, 8 bits peuvent être suffisants [Holi and Hwang, 1993].

L'implémentation de réseaux de neurones sur FPGA est un champ de recherche actif grâce à l'accélération que l'on peut obtenir avec un temps de développement raisonnable. Les applications concernent le plus souvent l'apprentissage numérique ou la simulation biologique, deux domaines ayant des contraintes très différentes. Certaines architectures ont eu un certain succès. L'architecture NeuroFlow par exemple est une architecture multiplexée compatible avec le cadriciel de description de réseaux de neurones python PyNN et qui permet d'implémenter sur un système multi-FPGA un réseau de taille importante (600000 neurones avec 6 FPGA) [Cheung et al., 2016]. L'approche utilisée est le multiplexage temporel afin de pouvoir implémenter différents modèles de neurones et des synapses STDP (spike-timing-dependent plasticity) en utilisant la RAM. Une approximation utilisée est l'apprentissage STDP avec les voisins les plus proches pour éviter une connexion complète des couches.

Une implémentation FPGA de réseaux impulsionnels Minitaur à calcul événementiel est capable d'accélérer les calculs des réseaux profonds appliqués à la base de donnée MNIST avec 22000 slices [Neil and Liu, 2014]. Là encore l'implémentation est multiplexée.

L'avantage de ces approches est une accélération des calculs de 2 à 10 fois le calcul GPU avec une réduction de consommation d'énergie par rapport au GPU ou CPU.

Les architectures parallèles sont souvent plus limitées en termes de nombre de neurones puisqu'elles se limitent à des MLP à quelque centaines de neurones.

Dans [Canals et al., 2016] les auteurs présentent une architecture parallèle à base de calcul stochastique. Cette architecture sans apprentissage repose sur un encodage (Extended Stochastic Logic, ESL en anglais) un peu plus complexe que les encodages stochastiques classiques puisque les réels sont encodés par un ratio de deux flux stochastiques. Cette approche a l'avantage de réduire le bruit dans l'architecture [Canals et al., 2015] et d'étendre l'encodage à tous les nombres fractionnels. Dans [Patel et al., 2007a] une architecture parallèle est présentée utilisant un encodage stochastique classique. Ils utilisent un accumulateur afin d'améliorer la qualité de l'opérateur d'addition et de soustraction.

Les implémentations à grain fin basées sur l'arithmétique stochastique ont donc moins de succès que les architectures multiplexées. Plus généralement, paralléliser à l'extrême comme nous l'étudions dans ce manuscrit est souvent mal vu pour plusieurs raisons. La première c'est que paralléliser a du sens uniquement lorsque la performance et le coût sont compétitifs. Souvent ce compromis se situe à un grain de parallélisme plus grossier. La seconde raison est que la parallélisation à grain fin ne permet pas de bénéficier des avantages et des progrès technologiques des micro-processeurs plus conventionnels [Omondi and Rajapakse, 2006].

Ce travail constitue cependant une recherche fondamentale sur les bénéfices possibles de ce type de parallélisme à grain fin pour la mise à l'échelle et la robustesse. L'intérêt n'est donc pas d'optimiser le coût ou la performance de l'implémentation des réseaux de neurones.

2.2 Implantations neuromimétiques

Le design neuromimétique est un paradigme de développement matériel démarré en 1990 par Carver Mead [Mead, 1990]. L'idée est de s'inspirer du remarquable pouvoir de calcul des systèmes neuro-biologiques. Malgré des unités de calcul lentes et bruitées, les neurones sont beaucoup plus performants que les ordinateurs les plus puissants pour les tâches de vision, d'audition et de contrôle moteur. Traditionnellement les ingénieurs neuromimétiques ont utilisé des neurones analogiques sur VLSI car ce sont les modèles les plus proches des neurones biologiques en termes de surface et de consommation énergétique [Mead, 1990]. C'est donc le meilleur candidat pour atteindre des tailles bio-réalistes. En s'inspirant de l'organisation des systèmes biologiques avec un substrat analogique asynchrone les ingénieurs neuromimétiques ont développé des systèmes d'acquisition bio-inspirés très performants tels que les rétines ou cochlées artificielles et autres [Indiveri and Horiuchi, 2011, Indiveri et al., 2011].

De nos jours, la définition des substrats neumimétiques s'est étendue à tous les substrats matériels (analogique, numérique ou mixte) qui permettent d'implanter des systèmes neuronaux. C'est à dire le développement de systèmes artificiels en utilisant les propriétés physiques et la représentation des informations des systèmes neuro-biologiques. L'idée fondamentale est de faire de l'ingénierie inverse du cerveau.

«As engineers, we would be foolish to ignore the lessons of a billion years of evolution» - Carver Mead 1993.

Un des grands succès des travaux neuromimétiques a été la mise en place d'un protocole de communication pour les neurones impulsionnels très performant inspiré du nerf optique.

Le nerf optique contient 1,2 million d'axones myélinisés le tout entouré d'une gaine méningée. C'est donc un canal haut débit qui permet de transmettre les impulsions des cellules ganglionnaires de la rétine vers les corps géniculés latéraux.

En 1992 des modèles de rétines artificielles existaient déjà sur VLSI mais il n'y avait pas de moyen efficace pour transmettre les informations à une vitesse suffisante aux autres circuits matériels. D'où l'introduction du protocole AER pour Adress Event Representation ou représentation par adresse d'événement [Mahowald, 1992] ; un bus de communication numérique qui sert d'intermédiaire de communication. Le principe est d'utiliser un seul bus avec un système d'accusé de réception pour prendre en compte chaque événement (impulsion) par un système de multiplexage temporel. Lorsqu'un neurone émet une impulsion il envoie une requête pour accéder au bus AER. Le bus transmet alors l'adresse du neurone activé à un multiplexeur qui informe les neurones ou synapses cibles. Un accusé de réception est alors transmis au neurone initiateur (voir fig 2.4).

La principale hypothèse de ce protocole est que le traitement des événements par le bus AER est plusieurs ordres de grandeur plus rapide que les constantes temporelles des neurones de sorte qu'il y ait rarement plusieurs événements demandant l'utilisation du bus au même moment. Par conséquent le bus transmet de façon quasi-instantanée (par comparaison avec les constantes temporelles des neurones) les événements. L'ordre des séquences d'activation est donc respecté.

Ce protocole est encore très utilisé de nos jours et de nombreuses extensions ont vu le jour afin d'améliorer son échelle et l'intégration des poids synaptiques. L'intégration d'une RAM (random access memory) permet d'associer les poids synaptiques plus efficacement dans [Benjamin et al., 2014], méthode ensuite développée en une approche de synapses partagées.

Il existe évidemment beaucoup d'autres approches pour acheminer les impulsions d'un réseau de neurones de taille élevée ([Vainbrand and Ginosar, 2011]). On peut noter par exemple le NoC multidiffusion (NoC pour Network on Chip ou réseau sur puce en français) qui est utilisé dans

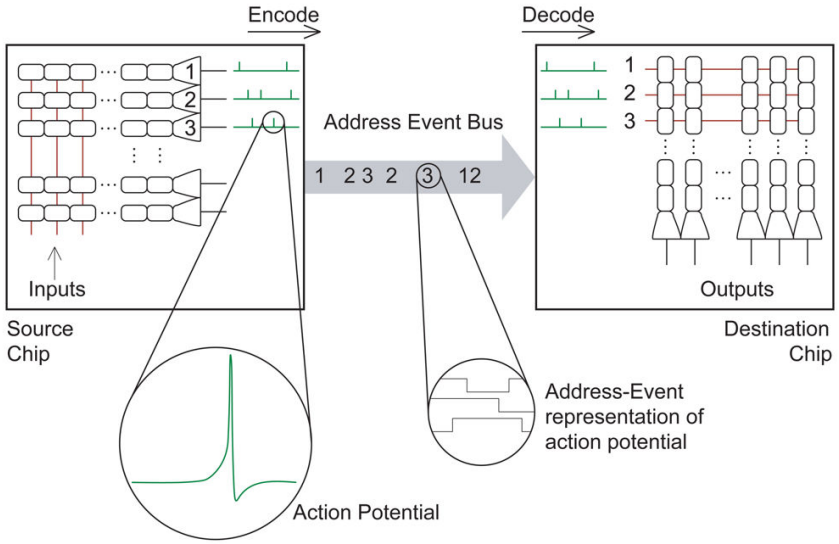


FIGURE 2.4 – Le protocole AER. Source : [Indiveri, 2008].

le projet SpiNNaker [5]. Ce projet devrait atteindre 10^9 neurones simulés dans peu de temps. La quantité d'événements à traiter est donc considérable, il serait impensable d'utiliser un seul bus. La plupart des ressources matérielles des cartes SpiNNaker sont donc utilisées par des routeurs qui dirigent les impulsions en fonction de l'adresse du neurone d'origine. C'est le principe de la multidiffusion.

Les travaux sur les architectures neuromimétiques ont aussi permis le développement de capteurs bio-inspirés communiquant avec le protocole AER : rétine artificielle [Lichtsteiner et al., 2008], cochlée artificielle [Sarpeshkar et al., 1998] et olfaction neuromimétique [Koickal et al., 2007] par exemple. Ces capteurs sont souvent implémentés sur des substrats analogiques VLSI, ont besoin d'une puissance faible et ont une bande passante réduite pour une fréquence d'acquisition très élevée. Le principe général est d'avoir une communication et des calculs déclenchés par des événements plutôt que par une horloge globale à fréquence fixe. Les caméras neuromimétiques par exemple envoient un événement quand la différence d'intensité d'un pixel dépasse un certain seuil. Ce seuil est localement adapté en fonction de la luminosité moyenne du pixel de sorte que le flux d'événements ne dépende pas de la luminosité et ce localement pour chaque pixel. Ceci est fait de manière asynchrone de sorte que la fréquence d'acquisition ne soit limitée que par la sérialisation du bus AER de sortie. Nous appellerons les architectures compatibles avec ce type d'architecture les architectures basées sur les événements même si elles ne sont pas forcément asynchrones.

2.3 Champs neuronaux dynamiques impulsionnels

Étant donnée la densité de connexions très importante dans le cas des champs neuronaux dynamiques (chaque neurone est connecté à tous les autres neurones), il n'est pas réaliste d'implémenter les neurones et les connexions en parallèle. Si l'on souhaite implémenter un réseau avec 64×64 neurones, il faudrait 64^4 connexions, soit près de 17 millions. Même en utilisant

5. http ://apt.cs.manchester.ac.uk/projects/SpiNNaker/project/

une arithmétique à flux de bits qui ne requiert qu'un seul bit pour communiquer cela n'est pas réaliste.

Nous avons au contraire cherché à centraliser les communications en s'inspirant des implémentations neuromimétiques qui ont beaucoup d'avantages. Nous avons donc cherché à développer une architecture basée sur les événements. Le plus grand avantage de ces architectures est la bande passante très réduite qui permet de très grandes vitesses de transfert. C'est justement ce qui nous intéresse au vu des forts besoins en communications latérales des DNF.

Cependant pour bénéficier de ces avantages, il est nécessaire de modifier le modèle des champs neuronaux dynamiques afin de le rendre impulsionnel.

Dans [Chevallier and Tarroux, 2008], il a été montré qu'un champ de neurones impulsionnels à fuite (leaky integrate and fire, LIF) était capable de reproduire les mécanismes d'attention visuelle tels qu'ils étaient décrits dans [Rougier and Vitay, 2006]. L'expérience a été reproduite et analysée dans [Vazquez et al., 2011] et montre des résultats supérieurs en termes de suivi de cible et de robustesse. Nous appelerons ce modèle DNF impulsionnel ou SDNF pour Spiking DNF en anglais.

Le modèle pour le potentiel membranaire de chaque neurone U_i du SDNF est décrit comme suit :

$$\tau \frac{du_i(t)}{dt} = -u_i(t) + \sum_j \omega_{ij} F(u_j(t)) + S(t) + h. \tag{2.4}$$

Quand le potentiel membranaire atteint le seuil θ une impulsion est émise et le potentiel est réinitialisé au potentiel de repos h. Les impulsions sont intégrées au potentiel membranaire avec cette equation :

$$u_i(t+dt) = \begin{cases} h & \text{si } u_i(t) \geq \theta \\ u_i + \tau \frac{\partial u_i}{\partial t} dt + I^{syn}(t) & \text{sinon} \end{cases} \tag{2.5}$$

où I^{syn} décrit les influences post-synaptiques sur le potentiel du neurone qui sont considérées comme instantanées (discrétisation temporelle d'un Dirac).

$$I^{syn}(t) = \sum_j \omega_{ij} I_j(t) \tag{2.6}$$

et

$$I_i(t) = \begin{cases} 1 & \text{si } U_i(t) \geqslant \theta \\ 0 & \text{si } U_i(t) < \theta. \end{cases} \tag{2.7}$$

Le potentiel est donc mis à jour de façon instantanée par les impulsions. La dynamique du modèle lors des premières itérations de calcul est visualisée sur la figure 2.5. On peut voir l'avantage de ce modèle en termes de communication. Les informations nécessaires au calcul des contributions latérales (les activations) sont binaires et peu nombreuses. Il est aussi possible d'avoir des informations binaires avec le modèle DNF standard en utilisant une fonction d'activation de type Heaviside. Mais avec la mise à zéro des potentiels membranaires lors de l'activation dans le modèle SDNF on introduit une fréquence d'activation qui dépend de τ et qui diminue forcément la quantité totale d'activation.

On souhaite maintenant vérifier que le modèle impulsionnel que nous avons introduit conserve les propriétés émergentes des champs neuronaux dynamiques. Nous allons pour cela utiliser le même cadre expérimental que dans le chapitre précédent afin de comparer le comportement des deux modèles dans le cadre du suivi de cible de la compétition et de la mémoire de travail reproduisant ainsi une partie des résultats obtenus dans [Vazquez et al., 2011].

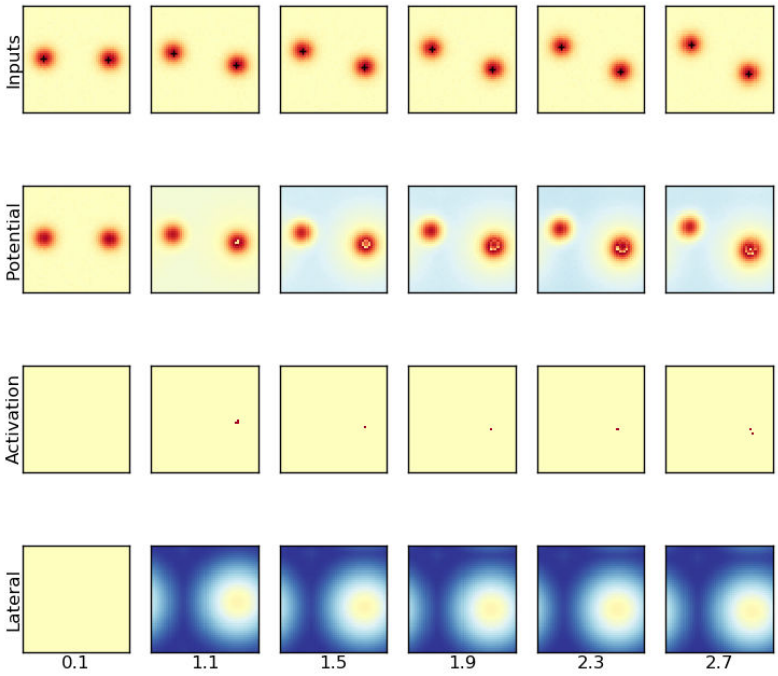


FIGURE 2.5 – Premières itérations d'un modèle DNF impulsionnel avec une constante temporelle élevée ($\tau = 0.7$) afin de mieux visualiser la dynamique. Les stimuli (première ligne) augmentent le potentiel synaptique des neurones du DNF jusqu'à la première activation provoquant une impulsion (troisième ligne). Les impulsions sont convoluées avec le noyau d'interaction latérale pour calculer les interactions latérales (dernière ligne).

TABLE 2.1 – Paramètres et leur intervalle utilisé pour l'optimisation du modèle de DNF impulsionnel

Paramètre	Intervalle	Compétition	Mémoire de travail
iExc	$[0, 5]$	0.658	0.343
wExc	$[0, 1]$	0.470	0.086
iInh	$[0, 5]$	0.645	0.240
wInh	$[0, 1]$	0.679	0.089
τ	$[0, 1]$	0.127	0.117
h	$[-1, 1]$	0.0	-0.017

Comme dans le chapitre précédent les paramètres du DNF impulsionnel sont trouvés avec l'algorithme des essaims particulaires avec les contraintes de la table 2.1. Des petites modifications sont nécessaires dans la fonction d'évaluation et sont décrites dans l'annexe A.

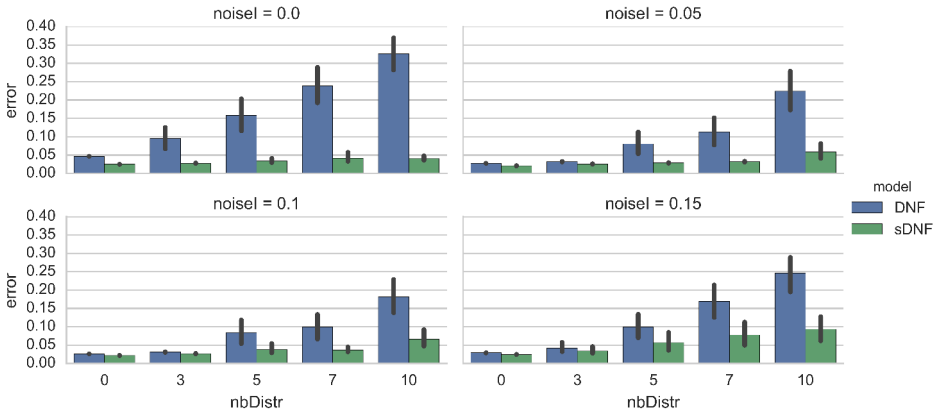
Sur la figure 2.6 on peut voir que le comportement du DNF impulsionnel est similaire à celui du DNF originel pour la compétition et la mémoire de travail.

On peut noter cependant une robustesse un peu plus grande pour le DNF impulsionnel ce qui rend ce modèle encore plus attractif. Cette plus grande robustesse peut venir de deux facteurs. Le premier est la nature impulsionnelle des interactions latérales qui permet leur intégration immédiate dans les potentiels membranaires. Le modèle est donc plus réactif. Le deuxième facteur est la remise à zéro du potentiel synaptique à chaque fois que le neurone est activé. Ceci permet de remettre à zéro les informations erronées que le neurone avait reçues et participe donc à l'amélioration du suivi.

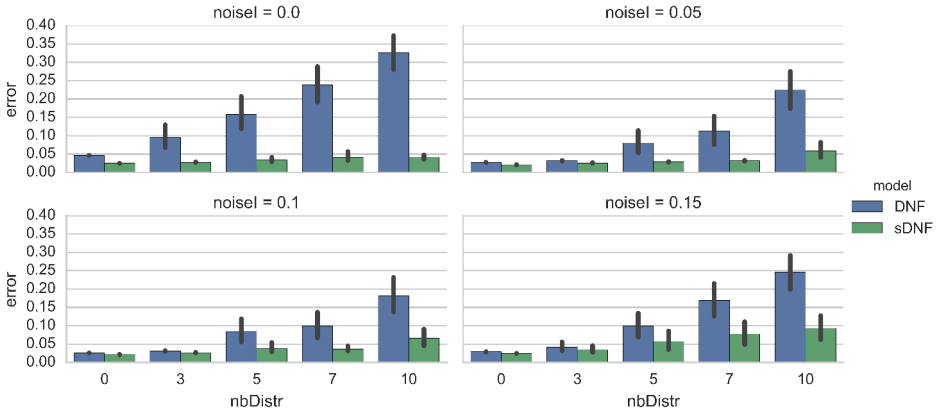
Il est important de noter que cette comparaison, même si elle est valide pour ce jeu de paramètres, ne démontre pas que le minimum global de la fonction fitness pour un des scénarios est moins bon que le minimum global des DNF impulsionnels. En revanche c'est un indice fort qu'il est plus facile de trouver des bons paramètres pour le DNF impulsionnel que pour le DNF classique. Il faut aussi noter que le DNF a un paramètre supplémentaire qui est le β de la sigmoïde ce qui pénalise la vitesse d'optimisation. Cependant, les résultats sont similaires si on utilise une fonction d'activation Heaviside.

2.4 Implémentation sur FPGA

Nous avons maintenant tous les outils pour pouvoir implémenter les champs neuronaux dynamiques de manière optimisée sur un substrat matériel numérique. Ceci répond à une demande qui existe pour l'accélération du calcul des champs neuronaux dynamiques ou de leur intégration dans des architectures matérielles développées sur FPGA. Ceci permet aussi d'avoir un référentiel de comparaison pour les architectures présentées dans les prochains chapitres. Le but de cette thèse est d'étudier les architectures cellulaires de DNF ; leur performance matérielle, bien que n'étant pas le but premier de cette étude, est un critère nécessaire car il permet de montrer la faisabilité d'une architecture dans un futur plus ou moins proche. La comparaison avec une architecture DNF optimisée permet d'avoir une idée des efforts restant à fournir pour aboutir à une architecture cellulaire véritablement exploitable.



(a) Compétition et suivi



(b) Mémoire de travail et suivi

FIGURE 2.6 – Erreur moyenne en fonction du nombre de distractions et du bruit. Comparaison du comportement du modèle DNF et DNF impulsionnel (sDNF).

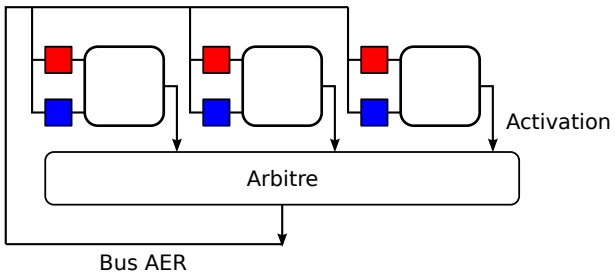


FIGURE 2.7 – Architecture AER pour les DNF impulsionnels. L'activation d'un neurone est prise en compte de façon non bloquante par l'arbitre AER. Lors des cycles de diffusion, l'arbitre publie successivement les coordonnées XY de chaque neurone activé dans l'étape de calcul précédente. Les synapses excitatrices en rouge et inhibitrices en bleu calculent alors le poids synaptique correspondant.

2.4.1 Implémentation parallèle

Plusieurs implémentations ont été proposées au cours de cette thèse pour les champs neuronaux impulsionnels avec un grain de parallélisme plus ou moins fin.

Chaque implémentation repose sur une communication des impulsions par l'intermédiaire d'un bus AER assurant une publication des neurones activés en utilisant comme identifiant leur position XY. Étant donné que la communication est souvent complète (connectivité all-to-all) chaque événement est publié à toutes les synapses. Le problème est donc simplifié vu qu'il n'y a pas besoin de table de routage. A la réception d'un événement il est ainsi possible de calculer la distance (de Manhattan) avec le neurone activé et on peut utiliser une table d'association plus réduite contenant uniquement les distances entre neurones possibles.

Nous avons fait une implantation FPGA de ce protocole de publication AER avec un modèle de neurone LIF simplifié [Chappet de Vangel et al., 2014] et montré la performance en termes de vitesse d'horloge et de surface d'implémentation. Cette implémentation n'utilise pas de multiplexage et est donc relativement coûteuse en termes de surface. Nous en décrivons ci-dessous les trois principaux composants.

2.4.1.1 Architecture d'un neurone LIF

Le but ici est d'implanter les neurones de la version discrète et bi-dimensionnelle de l'équation du LIF (voir section 2.3).

$$U_{x,y}(t+dt) = \frac{dt}{\tau}(-U_{x,y}(t) + h + S(t)) + I^{syn}_{x,y}(t) \tag{2.8}$$

et

$$I^{syn}_{x,y}(t) = De_{x,y}(t) - Di_{x,y}(t). \tag{2.9}$$

$De_{x,y}(t)$ et $Di_{x,y}(t)$ sont les contributions synaptiques des synapses excitatrices (resp. inhibitrices) qui sont données par les dendrites excitatrices (resp. inhibitrices). Afin d'optimiser le calcul en termes de temps et d'espace, $\frac{dt}{\tau} = 1/Fact$ sera pré-calculé de façon à ce que la division par $Fact$ soit réalisable par un simple décalage de bits (nous utilisons généralement $Fact = 8$ avec $dt = 0,08$ et $\tau = 0,64$).

Le calcul effectué par le module du neurone LIF est résumé sur le diagramme 2.8.

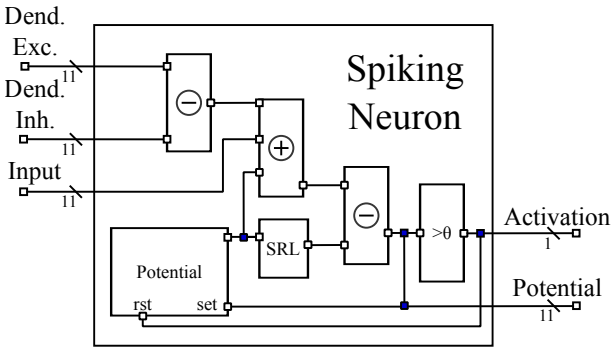


FIGURE 2.8 – Architecture des neurones LIF.

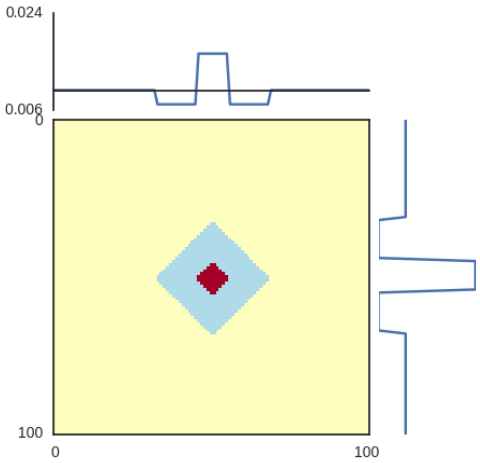


FIGURE 2.9 – Noyau latéral par parties, pour des paramètres correspondant à une mémoire de travail ($k_e = 1.0$, $k_i = 0.28$, $w_e = 0.05$, $w_i = 0.18$).

2.4.1.2 Implantation des synapses AER

Les synapses «écoutent» le bus AER, calculent la distance au neurone activé lorsque les coordonnées d'un neurone sont publiées et utilisent leur table interne (la même pour toutes les synapses excitatrices et la même pour toutes les synapses inhibitrices) pour mettre à jour leur potentiel post-synaptique. Lorsqu'un neurone met à jour son potentiel, il utilise les valeurs de sa synapse excitatrice et celle de sa synapse inhibitrice et les réinitialise.

Une alternative peut être proposée pour réduire un peu la surface des synapses. L'idée est d'utiliser une version simplifiée de la fonction des poids latéraux (fig. 2.9), une fonction par parties :

$$w(d) = \begin{cases} k_e - k_i & \text{si } d \leq w_e \\ -k_i & \text{si } w_e < d \leq w_i \\ 0 & \text{si } w_i < d. \end{cases} \tag{2.10}$$

Dans [Chappet de Vangel and Fix, 2016] nous avons montré qu'il était possible de trouver les paramètres de cette fonction avec une optimisation par essaims particulaires pour des scénarios de mémoire de travail et de compétition similaires à ceux présentés dans ce manuscrit. En utilisant ce

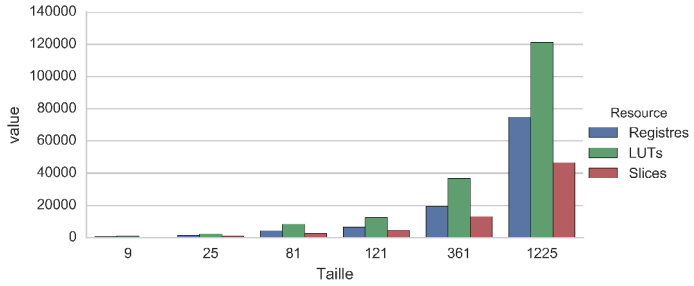


FIGURE 2.10 – Surface de l'implémentation parallèle des DNF.

noyau, il n'est plus nécessaire de sauvegarder les contributions latérales correspondant à chaque distance dans une mémoire associative mais on peut utiliser deux comparateurs. Le gain est donc important vu le nombre de synapses nécessaire.

2.4.1.3 Le bus et arbitre AER

Le bus AER contient un registre avec un bit pour chaque neurone. Les bits des neurones activés pendant le cycle de «calcul» sont à «1». Lors du cycle de «propagation» l'arbitre fait une recherche dichotomique pour trouver les bits activés, leurs associe la bonne coordonnée et publie cette coordonnée sur le bus de publication. Le bit du neurone est alors remis à «0».

Cette implémentation est donc complètement parallèle puisqu'elle alloue des ressources à chaque neurone. Le parallélisme synaptique est en revanche multiplexé temporellement puisqu'on utilise un bus commun pour toutes les impulsions. Cette approche permet d'obtenir une surface d'implémentation raisonnable (voir figure 2.10) mais dont la mise à l'échelle reste limitée pour ce qui est du nombre de neurones. Ceci est dû au nombre de LUT qui augmente rapidement.

2.4.2 Implémentation mixte

Une deuxième implémentation déporte le calcul neuronal sur un processeur multiplexant ainsi le calcul neuronal afin d'obtenir une plus grande précision et afin d'économiser de la surface (fig. 2.11).

Les synapses ne sont en revanche pas multiplexées et seront la principale limite à la mise à l'échelle. Les entrées/sorties et les communications intra/inter cartes sont toutes faites par l'intermédiaire du bus AER correspondant avec les coordonnées du neurone ou du pixel à l'origine de l'événement sur 15 bits : X 7 bits, Y 7 bits, valide 1 bit.

Les détails de l'architecture sont décrits dans les paragraphes et les figures suivantes. Un pseudo-langage proche du VHDL est utilisé pour faciliter la compréhension. Les constantes déterminant la taille des ports et leur signification sont explicitées dans la table 2.2.

2.4.2.1 Mise à jour des neurones

Les modules de mise à jour des neurones ont en réalité plusieurs fonctions. Chaque module de mise à jour (appelé un peu abusivement «processeur») est chargé d'un sous-ensemble de neurones pour lesquels un bloc de mémoire est associé.

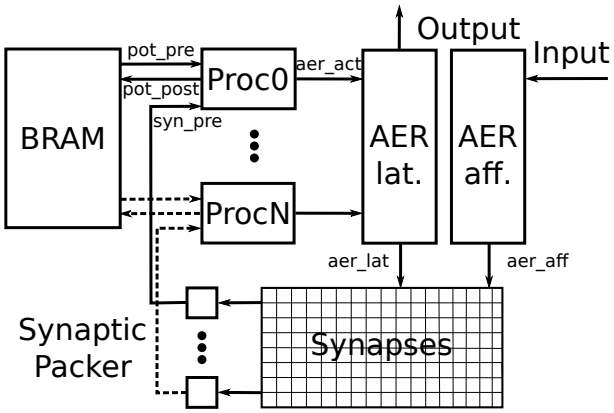


FIGURE 2.11 – Architecture utilisant les blocs mémoire du FPGA afin de multiplexer le calcul neuronal. Les événements d'entrées et latéraux sont intégrés dans la grille de synapses. Plusieurs processeurs parallèles effectuent la mise à jour du potentiel de chaque neurone avec une précision de 18 bits en lisant et en écrivant dans un bloc mémoire et en lisant les potentiels synaptiques.

TABLE 2.2 – Constantes utilisées pour l'implémentation du DNF impulsionnel de 64×64 neurones.

Constante	Valeur	Signification
R	64	résolution
P	18	taille du potentiel
S	16	taille des potentiels synaptiques
W	15	taille du bus AER
N	16	nombre de processeurs pour le calcul des potentiels
A	8	nombre de bits d'adresse pour stocker les potentiels

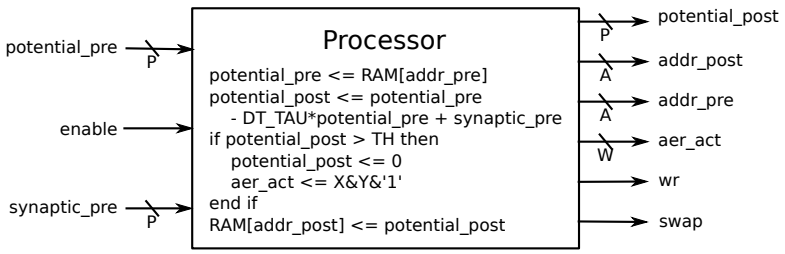


FIGURE 2.12 – Module de mise à jour des potentiels membranaires.

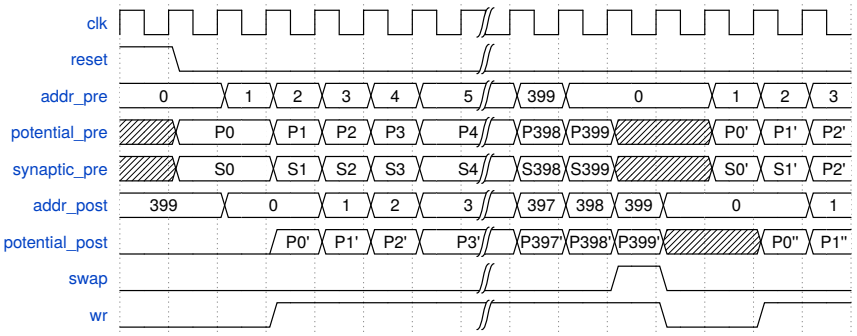


FIGURE 2.13 – Chronogramme du module de mise à jour du potentiel pour 400 neurones. Le signal *swap* permet l'échange des tampons synaptiques (fait avec Wavedrom).

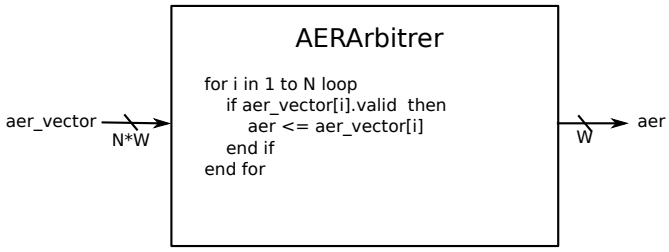


FIGURE 2.14 – Module pour l'arbitrage de plusieurs bus AER.

Il met à jour chacun de ces neurones séquentiellement à l'aide de plusieurs données. Le potentiel du neurone au temps t ($potential\_pre$) est récupéré à l'adresse $addr\_pre$ du bloc mémoire. Les contributions synaptiques sont récupérées auprès du module «SynapticPacker» qui associe simplement la bonne valeur synaptique à l'adresse présentée par le processeur.

Ces données permettent de calculer le nouveau potentiel membranaire $potential\_post$ et, si celui-ci est plus grand que le seuil d'activation, un événement valide est produit sur le port de sortie aer_act.

L'événement doit contenir les coordonnées absolues du neurone XY. Il est donc nécessaire que le processeur maintienne deux valeurs x et y pour chaque mise à jour. Il est possible d'assurer une mise à jour de potentiel par cycle d'horloge moyennant deux cycles d'horloge de latence (voir le chronogramme sur la figure 2.13).

Le module est aussi chargé de signaler aux synapses dépendantes la fin d'un cycle de calcul par le signal *swap*.

2.4.2.2 L'arbitre AER

Le module «AERArbitrer» est chargé d'arbitrer les événements aer_act provenant de tous les processeurs de l'architecture (fig. 2.14). Dans le cas (supposé très rare) où plusieurs processeurs sont activés en même temps, l'arbitre choisira un seul événement publié sur le bus AER de sortie.

2.4.2.3 Calcul des contributions synaptiques

Le calcul des contributions synaptiques de chaque neurone est pris en charge par un module «SynapticAccum». Cette partie de l'architecture est parallèle et prend donc la majorité de la

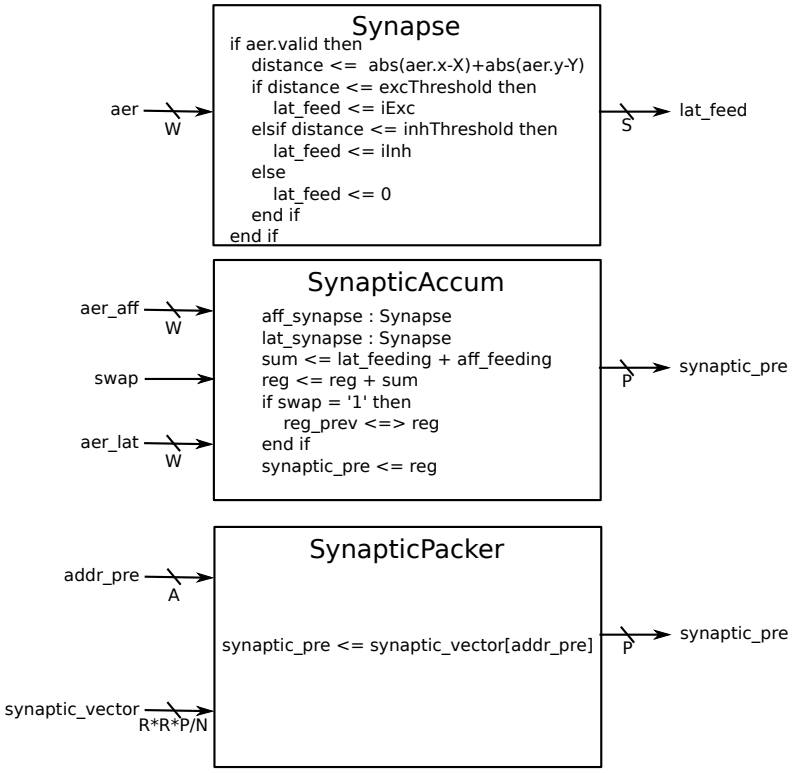


FIGURE 2.15 – Modules pour le calcul des contributions synaptiques.

surface. Ce module additionne toutes les contributions synaptiques reçues par le neurone et les stocke dans un registre *reg*. Lorsque le signal *swap* est haut, un échange de registres stocke cette valeur dans le registre reg_prev qui n'est pas modifié et qui est lu par le processeur (par l'intermédiaire du *synapticPacker*).

Le calcul des poids synaptiques est fait par les modules «Synapse» internes au module «SynapticAccum» et dédiés à chaque bus AER. Dans une architecture mono-carte par exemple, deux bus AER participent aux contributions synaptiques : un bus pour les stimuli (contributions afférentes) et un bus pour les contributions latérales. Par conséquent deux instances du module «Synapse» vont calculer les valeurs $aff\_synapse$ et $lat\_synapse$ qui seront additionnées avant d'être accumulées dans le registre *reg*.

2.4.2.4 Entrées-sorties

Les entrées-sorties transitent par l'intermédiaire d'un port USB en utilisant l'interface de transfert parallèle (DPTI) du circuit intégré FT2232H de la plate-forme de développement Digilent Nexys Video. Avec ce protocole il est possible de transférer en parallèle 8 bits de données avec une horloge de 60MHz ce qui permet d'utiliser le taux de transfert de 480Mbits/seconde du protocole USB 2.0.

Les événements de la caméra DVS sont envoyés via USB à une unité centrale qui se charge de diminuer la taille des paquets à 16bits en enlevant l'horodatage et de les envoyer vers le port micro-usb de la carte de développement. Les événements sont ensuite reconstruits et envoyés directement aux synapses.

Pour les sorties, tous les événements du bus AER_lat sont séparés en deux paquets de 8 bits et envoyés vers l'unité centrale pour horodatage et traitement.

2.4.2.5 Synchronisation

Les deux sources d'événements (latérale et afférente) pourraient en théorie avoir des horloges différentes qui poseraient des problèmes de synchronisation et nécessiteraient l'utilisation du protocole de poignée de main à quatre phases comme c'est traditionnellement le cas dans les architectures AER. Pour simplifier on utilise ici une seule horloge de 60Mhz pour le bus AER latéral et afférent.

Il faut quand même synchroniser l'architecture pour obtenir le dt que l'on veut utiliser. Le pas d'intégration est important pour savoir combien d'impulsions seront perdues (s'il est trop grand) et quel sera le nombre moyen d'événements afférents par cycle de calcul.

Par exemple s'il y a en moyenne 150 Keps (kilo événement par seconde) en sortie de la caméra DVS, on aura en fixant des mises à jour toutes les $dt = 0.1 \times 10^{-3}$ s, 14 événements par mise à jour en moyenne. Comme montré dans les paramètres de la table 2.2, on choisit d'utiliser un DNF de 64×64 neurones pour une entrée de 128×128 pixels et on parallélise les calculs avec 16 processeurs qui prennent donc chacun en charge 256 neurones. Ils doivent donc avoir une période de calcul de $0.1 \times 10^{-3}/256 = 3.9 \times 10^{-7}$. En utilisant une horloge de période 16.667×10^{-9} s, il faut activer le calcul tous les 24 cycles.

Sur le chronogramme 2.16 on peut voir un exemple de synchronisation des modules. Dans l'exemple du chronogramme, il y a un processeur qui met à jour deux neurones, le processeur est donc actif tous les 4 cycles d'horloges.

Le mieux serait en plus de décaler la mise à jour de chaque processeur ou de n'avoir qu'un seul processeur pour s'assurer qu'aucun événement d'activation ne soit perdu mais par souci de mise à l'échelle nous n'avons pas étudié ces possibilités. De même si la fréquence de calcul est

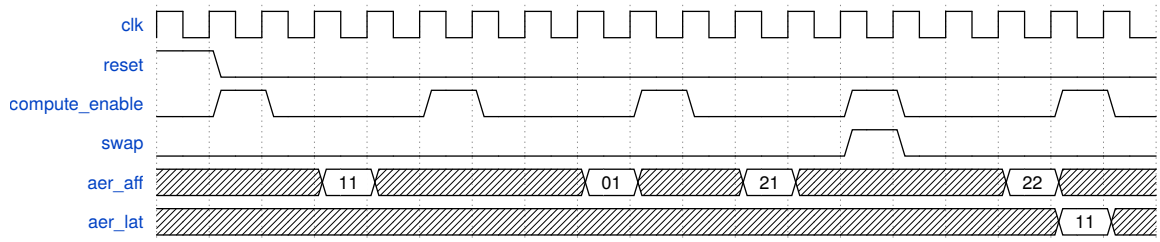


FIGURE 2.16 – Chronogramme de l'architecture globale. La mise à jour des potentiels est faite avec une certaine fréquence (1 cycle sur 4 ici) et si le processeur est en charge de deux neurones il faut attendre 4 mises à jour avant de passer au calcul suivant et inverser les tampons synaptiques. Les événement afférents arrivent à n'importe quel moment (via USB) et les événements latéraux ne peuvent apparaître qu'à partir du second cycle de calcul (fait avec Wavedrom).

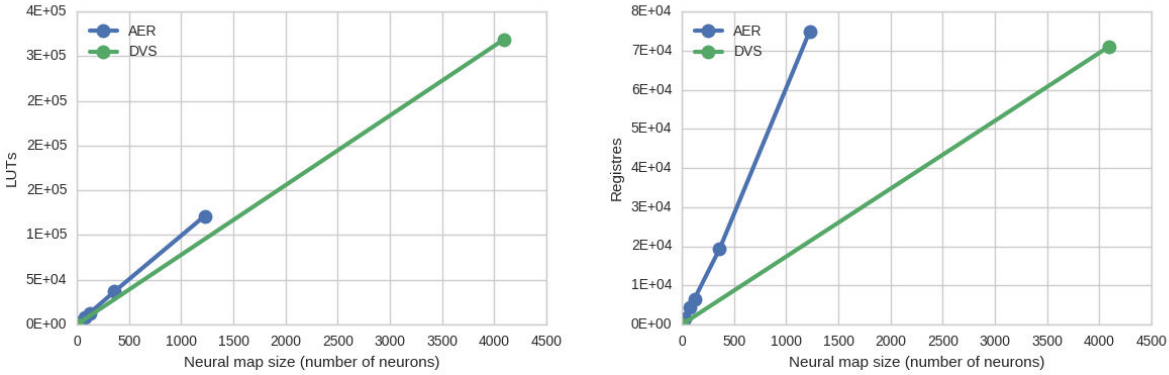


FIGURE 2.17 – Résultats d'implantation pour le modèle AER (parallèle) et DVS (mixte).

suffisement rapide, on pourrait postuler qu'il n'y a qu'un seul événement par synapse et par cycle de mise à jour et qu'il est donc inutile de les accumuler.

L'implantation de ce modèle sur FPGA Xilinx XC6VLX760T-3FF1156 montre une économie de surface par rapport à une implémentation parallèle pure (fig. 2.17). L'économie la plus importante est au niveau des registres puisque la sauvegarde de l'état des neurones est déportée sur le bloc mémoire. En revanche on voit que le calcul synaptique nécessite toujours beaucoup de LUTs malgrès l'introduction du noyau latéral pas partie.

Cette architecture est suffisamment générique pour connecter plusieurs cartes DNF sur le même circuit FPGA ou par d'autres interfaces via une communication par bus AER. Par exemple dans [Chappet de Vangel et al., 2016] nous avons utilisé cette architecture pour implémenter deux champs neuronaux en interaction dans un scenario d'exploration de scène. Les entrées externes sont reçues par USB depuis la caméra DVS128.

Plusieurs pistes existent pour améliorer cette architecture. La première est d'utiliser le multiplexage au niveau des synapses également. En fonction de la fréquence de l'horloge et de la

fréquence des événements, il parait réalisable de stocker les potentiels synaptiques dans un bloc mémoire et de les mettre a jour de façon séquentielle lorsqu'un événement est reçu.

Une autre approche rapprochant encore plus les calculs de ceux effectués dans la version logicielle est d'utiliser des IP pour faire des convolutions FFT par exemple, technologies devenant de plus en plus fréquentes après l'intérêt porté aux réseaux profonds convolutionnels.

* * *

Ce chapitre nous a permis de nous familiariser avec les enjeux des implémentations de réseaux de neurones sur un matériel numérique comme les FPGA. Nous avons vu la différence fondamentale qu'il y avait entre les implémentations parallèles et séquentielles et nous avons vu comment les chercheurs se sont inspirés de la neurobiologie pour concevoir des architectures plus rapides et consommant moins d'énergie dans le cadre de l'ingénierie neuromimétique. Ces réflexions ont menées à la définition d'un nouveau modèle de DNF : les DNF impulsionnels qui ont l'avantage d'avoir des besoins en communications beaucoup plus faibles. L'utilisation d'un bus partagé pour le transfert de ces informations nous a permis d'implémenter efficacement les DNF malgré le besoin de communication globale de chaque neurone. Nous avons ainsi pu concevoir des architectures parallèles à grains plus ou moins fins intégrées à des architectures événementielles basées sur des capteurs neuromimétiques comme la rétine artificielle.

Cependant, ces architectures ont les mêmes défauts que les architectures de von Neumann. La mémoire est séparée des unités de calcul et la communication entre les deux reste le facteur limitant pour la vitesse et la mise à l'échelle. Le bus AER même s'il permet une communication à grande vitesse, limite aussi la mise à l'échelle de l'architecture par sa bande passante.

La robustesse aux fautes matérielles est elle aussi compromise par l'existence de ces voies de communication centralisées. Si une faute compromet le fonctionnement de ces routes, toute l'architecture sera compromise.

Il est donc nécessaire d'explorer de nouveaux types architectures afin d'obtenir un substrat matériel avec des propriétés de mise à l'échelle et de robustesse davantage compétitives en comparaison avec le cortex des mammifères.

Chapitre 3

Champs neuronaux dynamiques impulsionnels aléatoires

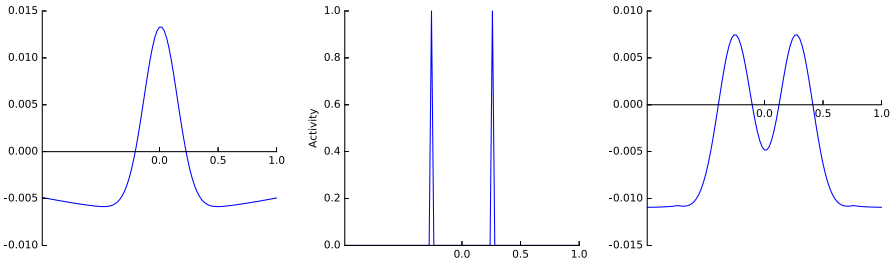
Nous avons vu dans l'introduction que le matériel de calcul ciblé est le substrat naturel. Nous avons vu les propriétés de ce substrat et pourquoi elles sont désirables pour un substrat de calcul. Cependant ces propriétés émergent d'interactions à beaucoup d'échelles différentes. Si l'on souhaite avoir un matériel régénératif par exemple, il faut se placer au niveau des cellules biologiques formant des tissus biologiques aux propriétés régénératives. Si l'on souhaite avoir un matériel avec la capacité de calcul du cortex il faut se placer au niveau des neurones ou des populations de neurones (comme nous le faisons dans cette thèse) et l'interaction de ces unités permet alors d'obtenir un matériel avec des propriétés de calcul bio-inspirées. En revanche si l'on veut un matériel résistant aux fautes il faut se placer au niveau physico-chimique, substrat de base du vivant qui a de très bonnes propriétés de résistance aux fautes. On voit donc qu'il y a un problème d'échelle. Il parait difficile d'obtenir toutes les propriétés désirées sans modéliser tous les niveaux depuis le substrat primaire, ce qui n'est pas réaliste pour des applications embarquées à cause des ressources de calcul nécessaires.

Dans ces travaux nous cherchons à palier à ce problème en mélangeant plusieurs niveaux de simulation afin de profiter de plusieurs propriétés naturelles. On va voir que cela nécessite d'identifier l'essence des dynamiques recherchées afin de pouvoir au besoin modifier les modèles sous-jacents et ainsi faciliter l'émergence de toutes les propriétés voulues.

Les propriétés que nous recherchons sont pour l'instant assez restreintes. Le substrat doit reproduire la dynamique des champs neuronaux dynamiques, il doit être robuste aux fautes et la mise à l'échelle doit être réalisable. Dans le futur, il est possible de chercher à ajouter des propriétés d'apprentissage mais pour l'instant ce n'est pas notre objectif.

Pour avoir toutes les propriétés recherchées nos recherches se sont orientées vers un substrat cellulaire pour différentes raisons. La première raison est que le calcul cellulaire garantit une mise à l'échelle facile. La deuxième est qu'il parait plus aisé d'avoir une robustesse aux fautes dans ce type de substrat vu qu'il n'est pas centralisé et que les communications sont locales.

Le calcul cellulaire tel qu'il est défini par Sipper [Sipper, 1999] est un calcul à base d'unités relativement simples avec des interactions locales. La propriété de l'ensemble est alors une propriété émergente d'un grand nombre de ces cellules. De plus, la compatibilité naturelle avec le substrat de calcul numérique que nous utilisons. En effet le graphe de connectivité entre les blocs de calcul dans un FPGA a certaines limites de densité, il est donc souhaitable que cette limite ne soit pas atteinte quel que soit le nombre d'unités de calcul. Avec un calcul cellulaire au voisinage limité, il est garanti que la connectivité ne sera pas un frein à la mise à l'échelle. On



(a) Poids latéraux (b) Activité (c) Convolution

FIGURE 3.1 – Le calcul réalisé par les connections latérales du champs neuronal dynamique impulsionnel peut être ramené à une convolution de l'activité par la fonction des poids latéraux. a) exemple de fonction des poids latéraux de type différence de gaussiennes. b) exemple d'activité neuronale. Ici il y a deux neurones activés. c) résultat de la convolution de l'activité par les poids latéraux avec des bordures cycliques.

voit donc que les propriétés recherchées sont réalisables sur un substrat cellulaire.

3.1 Enjeux et méthodologie

3.1.1 Formalisation du problème

Nous allons considérer qu'un modèle de calcul est valide lorsque son comportement est meilleur ou équivalent au modèle de DNF pour tous les scénarios que nous avons étudiés dans la première partie (sélection, suivi de cible, suivi robuste et mémoire de travail).

Les interactions latérales telles qu'elles sont définies dans le modèle d'Amari avec une fonction d'activation de type Heaviside ou par la version impulsionnelle peuvent être ramenées au calcul de la convolution de l'activité binaire des neurones par le noyau d'interaction latérale qui est une différence de gaussiennes.

Nous cherchons donc à obtenir une approximation, par un calcul local, de l'équation suivante :

$$I^{lat}_{x,y}(t) = \sum_{x'} \sum_{y'} w(||x - x', y - y'||) F[u_{x',y'}(t)] \tag{3.1}$$

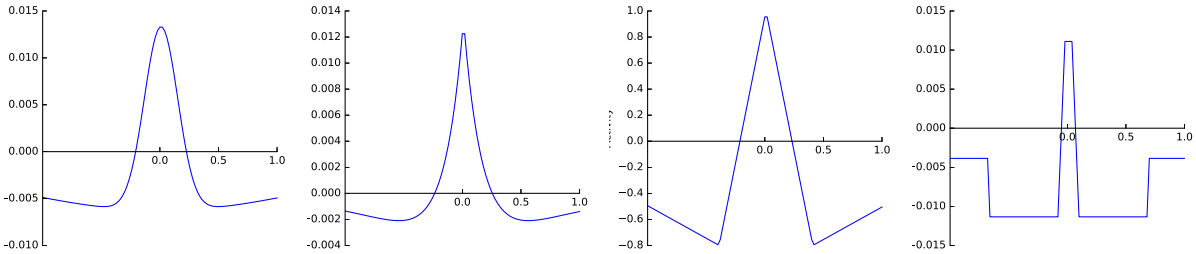
Où $||x - x', y - y'||$ est la distance euclidienne entre les position (x, y) et (x', y'). w est un noyau symétrique de type différence de gaussiennes.

On peut visualiser en une dimension le calcul à réaliser sur la figure 3.1.

3.1.2 Fonction des poids latéraux possibles

Comme Amari le note dans son papier [Amari, 1977], il n'est pas nécessaire que le noyau soit strictement une DoG, les conditions nécessaires et suffisantes sont :

1. $w(-x) = w(x)$
2. $w(x) > 0, \forall x \in [0, x0)$ avec $w(x_0) = 0$
3. $w(x) < 0, \forall x \in (x_0, \infty)$
4. $w(x)$ est décroissant sur $[0, x_0]$



(a) Gaussien (b) Exponentiel (c) Linéaire (d) Par partie

FIGURE 3.2 – Différents types de fonctions de poids latéraux étudiées.

5. $w(x)$ a un seul minimum sur $\mathbb{R}^+$ à $x = x_1$ avec $x_1 > x_0$ et $w(x)$ strictement croissant sur (x_1, ∞)

Le noyau doit donc être symétrique, positif décroissant dans la partie excitatrice centrale (de x à $x0$) et négatif sur le reste de la courbe avec un minimum global à $x = x1$ et croître ensuite. Nous allons utiliser cette liberté dans nos implantations matérielles vu qu'aucune implantation n'aura un noyau gaussien.

Voici quelques courbes utilisées pour la modélisation des poids latéraux sur la figure 3.2. Malgré les différences, nous pouvons voir que mise à part la dernière elles respectent les propriétés d'Amari.

3.1.3 Validation des modèles

Le développement des modèles est contraint par beaucoup de paramètres et obéit toujours aux mêmes étapes de développement :

1. Conception et formalisation du modèle matériel.
2. Étude et validation du noyau d'interactions latérales simplifié émergent. Il doit satisfaire tous les scenarii.
3. Modélisation des conditions limites de ce noyau et validation expérimentale.
4. Implémentation matérielle, tests et étude de la mise à l'échelle.
5. Analyse de la robustesse matérielle.

Afin de faciliter la lecture, les deux modèles développés au cours de cette thèse seront décrits séparément en suivant ces étapes. Nous étudierons ensuite de façon conjointe ces modèles afin de comparer leur performances, leur mise à l'échelle et leurs qualités cellulaires (robustesse, décentralisation etc.).

Tous les modèles présentés dans ce chapitre et le suivant sont des contributions originales de cette thèse et sont publiés.

3.1.4 Niveaux de simulation

Les différentes étapes de l'étude s'accompagnent de différents outils logiciels permettant de simuler les modèles à différentes échelles. Tous les niveaux sont regroupés dans un cadriciel écrit

en Python et lié à une bibliothèque C++ [6]. Nous parlerons de plusieurs niveaux de simulation dans cette partie.

1. Niveau asymptotique. La fonction des poids latéraux émergente est approchée par un noyau de convolution.
2. Niveau approché. La propagation des informations dans le graphe de connectivité est simulée en une seule itération de calcul.
3. Niveau cellulaire. L'architecture matérielle est implémentée telle quelle dans le simulateur à l'aide de registres synchrones. Chaque étape de calcul propage les informations d'un arc uniquement.
4. Simulation matérielle. Même simulation que la précédente mais la précision des registres est prise en compte dans les calculs. Cette simulation est supposée identique à l'exécution sur FPGA.
5. Exécution sur FPGA avec lecture de fichier pour les entrées et écriture de fichier pour les sorties.

Cette pile de développement logicielle est un peu lourde mais elle permet d'obtenir une vitesse de simulation maximale pour chaque étape de développement d'un modèle. Il est envisageable de simplifier cette pile logicielle par l'utilisation d'une bibliothèque de simulation matérielle haut niveau telle que SystemC [7].

3.2 Modèle des champs neuronaux impulsionnel aléatoires

Nous avons introduit les champs neuronaux dynamiques impulsionnels aléatoires (randomly spiking dynamic neural fields RSDNF en anglais) dans le but d'améliorer l'implémentation matérielle des DNF impulsionnels en enlevant la contrainte de graphe complet [Chappet de Vangel et al., 2015a]. En effet, dans le modèle SDNF, chaque impulsion est intégrée avec un poids synaptique dépendant de la distance entre le neurone source et le neurone cible avec une forme de DoG.

Par conséquent, la distance à l'origine est la seule information nécessaire lors de l'intégration d'une impulsion. On va ici remplacer cette information nécessitant une population de neurones complètement connectés. Ainsi l'architecture matérielle sera implémentée sur une grille simple où les neurones sont connectés uniquement avec leur quatre voisins. Dans cette grille nous allons remplacer les informations sur la distance entre les neurones pré-synaptique et post-synaptique par une propagation aléatoire d'impulsions de proches en proches permettant de diminuer la quantité d'impulsions reçues en fonction de la distance, reproduisant ainsi la forme de la DoG.

Lorsqu'un neurone est activé, il émet N qImpulsions excitatrices et N qImpulsions inhibitrices qu'il transmet directement à ses 4 voisins avec une probabilité p_a et p_b respectivement sur deux couches de routage différentes. Les qImpulsions sont des quanta d'impulsion. Les voisins vont transmettre à leurs tour les qImpulsions à leurs voisins en utilisant un tirage aléatoire avec la même probabilité de succès (voir figure 3.4). A chaque réception d'une qImpulsion, le neurone augmente son potentiel synaptique de ω_a si la qImpulsion est excitatrice et le neurone diminue son potentiel de ω_b si la qImpulsion est inhibitrice.

3.2.1 Routage

Le but de la grille de routage est de transmettre les informations d'un neurone activé à tous les autres neurones de la carte. Ici les informations n'ont pas besoin d'être emballées dans un

6. code source accessible sur https ://github.com/bchappet/dnfpy
7. https ://en.wikipedia.org/wiki/SystemC

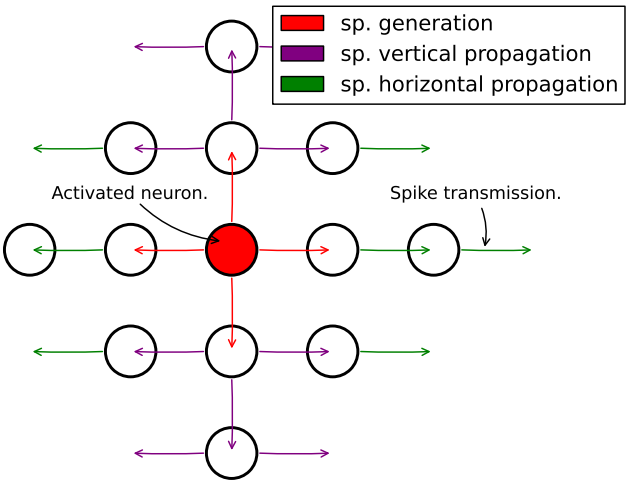


FIGURE 3.3 – Transmission des qImpulsions par une couche de routage de type diffusion XY. Le neurone activé (en rouge) charge ses quatre routeurs avec N paquets. Ces paquets sont ensuite transférés horizontalement par les routeurs horizontaux (flèche verte) ou bien transmis horizontalement et verticalement par les routeurs verticaux (flèche violette). Ainsi chaque neurone (cercles) différent du neurone activé recevra exactement N paquets sous la forme d'un bit.

paquet vu que tous les neurones sont ciblés de la même manière.

La propagation des qImpulsions est faite par un routage que l'on appellera *diffusion XY* car il est basé sur les algorithmes de routage XY [Palesi and Daneshtalab, 2014]. Le principe de cette diffusion est de dupliquer les paquets à chaque transfert vertical pour les transmettre horizontalement. Si un paquet est reçu d'une direction horizontale il sera transmis dans la même direction. En revanche, si une impulsion est reçue depuis une direction verticale, elle sera transmise dans la même direction ainsi que dans les deux directions horizontales (voir figure 3.3). Ce mode de diffusion requiert 4 routeurs pour chaque neurone correspondant aux quatre directions de transmission nord, sud, est et ouest.

De cette façon il est possible de transmettre les qImpulsions à tous les autres neurones avec un chemin unique et sans redondance. Le temps de propagation maximal est proportionnel à la distance de Manhattan maximale entre deux neurones de la carte neuronale c'est à dire $2R$ pour une carte de $R \times R$ neurones.

3.2.2 Formalisation

Soit une carte de neurones de taille $R \times R$. Chaque neurone est identifié par ses coordonnées i et j. Chaque neurone contient quatre routeurs excitateurs r^a_{ijk} et quatre routeurs inhibiteurs r^b_{ijk}, où $k \in \{0, 1, 2, 3\}$ identifie la direction du routeur par rapport à son neurone dans l'ordre nord, est, sud, ouest.

On définit les connexions entre les routeurs $r_{i_1 j_1 k_1}$ et $r_{i_2 j_2 k_2}$ avec des arcs de probabilité p.

$A^a_{\{i1,j1,k1\}\to\{i2,j2,k2\}} \in [0,1]$. Pour la couche excitatrice :

$$A^a_{\{i1,j1,k1\}\to\{i2,j2,k2\}} = p_a \quad \text{si} \begin{cases} i_2 = i_1 + 1, & j_2 = j_1, k1 = 0, & k_2 \in \{0,1,3\} \\ ou & & \\ i_2 = i_1, & j_2 = j_1 + 1, k1 = 1, & k_2 = 1 \\ ou & & \\ i_2 = i_1 - 1, & j_2 = j_1, k1 = 2, & k_2 \in \{1,2,3\} \\ ou & & \\ i_2 = i_1, & j_2 = j_1 - 1, k1 = 3, & k_2 = 3 \end{cases} \tag{3.2}$$

$$= 0 \quad \text{sinon} \tag{3.3}$$

Et de même pour la couche inhibitrice pour $A^b_{\{i1,j1,k1\}\to\{i2,j2,k2\}} = p_b$ ou 0.

Dans la suite on allègera certaines notations (en omettant par exemple les indices i et j lorsque cela est suffisamment implicite) afin de simplifier les équations.

L'ensemble des mécanismes de mise à jour des potentiels synaptiques u_{syn} des neurones peut être décrit comme suit :

Soit δ^t_k la variable aléatoire du nombre de qImpulsions stockées par un routeur r à l'instant t. Sachant que les qImpulsions sont traitées les unes après les autres, l'accumulation et le "dépilement" des qImpulsions stockées donne :

$$\delta^{t+1}_r = min(0, \delta^t_r - 1) + \gamma^t_r + NI^t. \tag{3.4}$$

I_t représente l'activation du neurone à l'instant t ($I^t = 1$ si le neurone est activé $u^t \geqslant \theta$, $I^t = 0$ sinon) et N est le nombre de qImpulsions générées lors d'une activation. γ^t_r est le nombre de qImpulsions reçues par un routeur depuis ses prédécesseurs :

$$\gamma^t_r = \sum_{\rho \in pred(r)} min(1, \delta^t_\rho) Ber(A_{\rho\to r}) \tag{3.5}$$

Où Ber est la distribution de Bernoulli et $pred(r)$ donne les prédécesseurs directs du routeur r.

On met alors à jour le potentiel synaptique du neurone

$$u^t_{syn} = \omega_a \sum_{k=0}^{k=3} \gamma^{a,t}_k - \omega_b \sum_{k=0}^{k=3} \gamma^{b,t}_k, \tag{3.6}$$

Le potentiel global du neurone est mis à jour avec l'équation du neurone LIF :

$$u^{t+1} = \begin{cases} 0 \text{ si } I^t = 1 \\ u^t + \dfrac{dt}{\tau}(-u^t + h + s^t) + u^t_{syn} \text{ sinon} \end{cases} \tag{3.7}$$

3.3 Poids latéraux

Nous allons ici formaliser la fonction des poids latéraux résultante de ce modèle en négligeant plusieurs facteurs. a) Nous allons d'abord considérer un contexte asymptotique avec $N \to \infty$. b) Nous allons ensuite négliger le temps de propagation des impulsions. C'est à dire que l'on attend que toutes les impulsions soient arrivées au bout de la carte de neurones avant de calculer

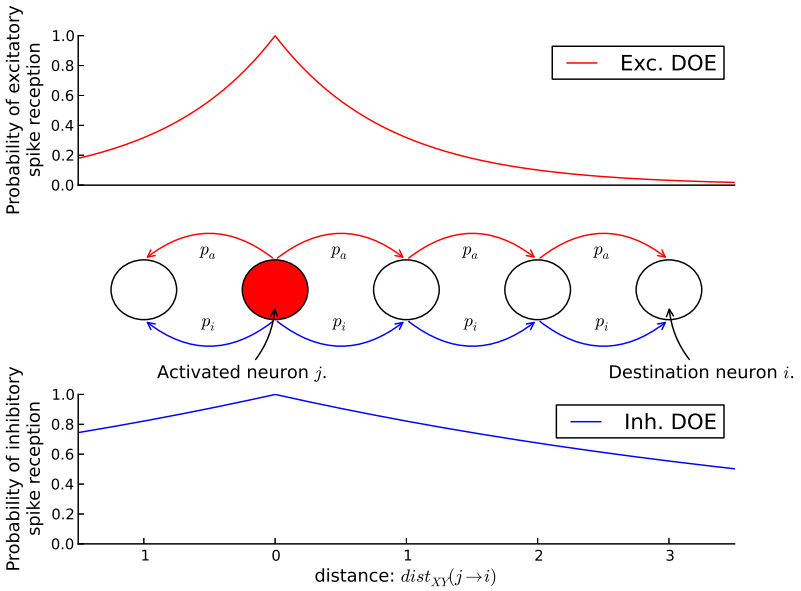


FIGURE 3.4 – Transmission des qImpulsions.

les nouveaux potentiels membranaires des neurones et de potentiellement générer de nouvelles activations.

Le but de cette étude est de vérifier que la fonction des poids latéraux vérifie bien les conditions nécessaires à la dynamique du champ de neurones et d'obtenir des paramètres permettant de réaliser les scénarios étudiés dans la première partie.

3.3.1 Formalisation

Nous sommes en conditions asymptotiques :

$$Ber(p) \overset{N\to\infty}{\longrightarrow} p \tag{3.8}$$

Par conséquent en considérant une transmission infiniment rapide des impulsions on a

$$\delta_r = \sum_{\rho \in Pred(r)} I_\rho N p^{d(\rho,r)}. \tag{3.9}$$

Où $Pred$ représente *tous* les routeurs prédécesseurs (directs et indirects) du routeur r, I_ρ représente l'activation du neurone associé au routeur ρ et $d(\rho, r)$ est la distance de Manhattan entre le routeur ρ et le routeur r (ou nombre d'arcs entre les routeurs ρ et r).

On peut donc écrire le potentiel synaptique du neurone xy :

$$u_{syn}[x,y] = w_a \sum_{z=0}^{z=3} \sum_{r_{ijk} \in Pred(r_{xyz})} I_i N p_a^{d(r_{ijk}, r_{xyz})} - w_b \sum_{z=0}^{z=3} \sum_{r_{ijk} \in Pred(r_{xyz})} I_i N p_b^{d(r_{ijk}, r_{xyz})}, \tag{3.10}$$

Le routage XY nous assure qu'il existe un unique chemin entre deux neurones. Ceci est dû au fait que pour le neurone x, y,

$$\bigcup_k Pred(r_{xyk}) = \mathcal{S} \tag{3.11}$$

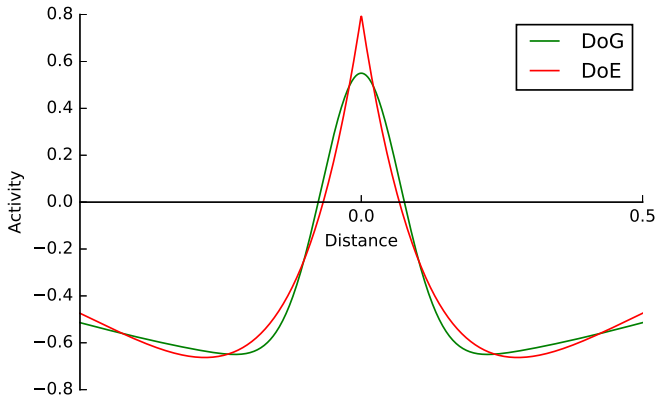


FIGURE 3.5 – Alignement de la DoE avec la DoG.

où $\mathcal{S}$ est l'ensemble des neurones et

$$\bigcap_k Pred(r_{xyk}) = \varnothing \tag{3.12}$$

On peut donc réécrire l'équation précédente en prenant en compte la dimension spatiale. Vu que tous les neurones sont prédécesseurs (au sens large) de chaque neurone on peut écrire :

$$u_{syn}[x,y] = \sum_i \sum_j w_a I_{ij} N p_a^{d(xy,ij)} - w_b I_{ij} N p_b^{d(xy,ij)}, \tag{3.13}$$

ou

$$u_{syn}[x,y] = \sum_i \sum_j I_{ij} w(d(xy,ij)), \tag{3.14}$$

en posant

$$w(d) = w_a N p_a^d - w_b N p_b^d \tag{3.15}$$

Ce qui correspond à la convolution de l'équation du potentiel synaptique qu'on a vue lors de la définition des champs neuronaux dynamiques impulsionnels avec un noyau d'interaction synaptique différent.

Nous allons maintenant étudier ce nouveau noyau d'interactions latérales que nous appellerons différence d'exponentielles (DoE pour difference of exponentials en anglais).

3.3.2 Alignement des courbes

Nous pouvons trouver une approximation des paramètres nécessaires à la simulation du DNF avec ce noyau en alignant la DoE avec la DoG. Afin d'obtenir des paramètres indépendants du nombre d'impulsions générées lors de chaque activation N nous posons, $\omega_a = N w_a$ et $\omega_b = N w_b$. On étudie donc la fonction

$$w(d) = \omega_a p_a^d - \omega_b p_b^d \tag{3.16}$$

Les paramètres trouvés lors de l'alignement des courbes (voir table 3.1) sont obtenus sur un espace continu entre $[-0.5, 0.5[$, espace de taille une unité. Si on discrétise cet espace sur R cellules, il faut adapter les paramètres de façon à ce que l'aire de la courbe soit identique.

TABLE 3.1 – Paramètres obtenus lors de l'alignement des courbes exponentielles avec les courbes gaussiennes.

Paramètre	Valeur	Description
Ω_a	250.54	intensité des poids excitateur.
Ω_b	249.74	intensité des poids inhibiteur.
p_a	0.0084	probabilité de propagation des impulsions excitatrice.
p_b	0.0088	probabilité de propagation des impulsions inhibitrice.

Démonstration. On cherche donc à trouver une relation entre p, ω et p', ω', R de façon à ce que :

$$\int_0^1 \omega p^x dx = \int_0^R \omega' p'^x dx$$

$$\iff \omega \frac{p-1}{ln(p)} = \omega' \frac{p'^R - 1}{ln(p')}$$

On pose $\omega' = \frac{\omega}{R}$ et $p' = p^{\frac{1}{R}}$

$$\underset{\omega' = \frac{\omega}{R}}{\overset{p' = p^{\frac{1}{R}}}{\iff}} \omega \frac{p-1}{ln(p)} = \frac{\omega(p-1)}{Rln(p^{\frac{1}{R}})}$$

□

On peut donc simuler le comportement du DNF avec ce nouveau noyau d'interactions latérales en utilisant la résolution de discrétisation souhaitée et les paramètres trouvés lors de l'alignement des courbes. Il s'avère que le comportement n'est pas correct, probablement à cause de la différence assez marquée entre les deux courbes pour les petites distances (voir figure 3.6).

3.3.3 Résultats

Pour comparer les performances de la DoE au autres noyaux il est préférable d'optimiser les paramètres pour chaque scénario comme nous l'avons fait dans les chapitres précédents. On voit sur la table 3.2 qu'il a été possible d'optimiser le noyau de type différence d'exponentielles afin d'obtenir un comportement similaire au noyau gaussien. Les intervalles d'optimisation ont été adaptés pour la mémoire de travail car seul des noyaux très localisés sont capables de faire émerger le comportement attendu.

Sur la figure 3.7 on peut voir la dynamique du modèle lors d'un scénario de suivi de cible robuste. Les impulsions sont visibles dans la ligne «Activation» mais l'influence du noyau de type DoE est visible dans la dernière ligne, celle des interactions latérales. On voit que l'utilisation de la distance de Manhattan pour le noyau induit des interactions latérales en forme de diamant. Néanmoins cela ne gène pas la dynamique puisque le suivi robuste de cible est toujours possible ainsi que la mémorisation.

On voit par exemple sur la figure 3.8 que les performances de la DoE sont similaires aux autres modèles. Les paramètres trouvés pour le suivi robuste de cible semblent un peu plus sensibles dans le cas de fort bruit et d'un grand nombre de distractions. En revanche les résultats sont plus robustes dans le cas de la mémoire de travail.

Cependant, ces résultats sont basés sur une approximation du comportement réel du RSNDF basé sur deux hypothèses : (a) N tend vers l'infini (b) la propagation des impulsions est immédiate. Nous allons maintenant étudier le modèle de façon plus réaliste.

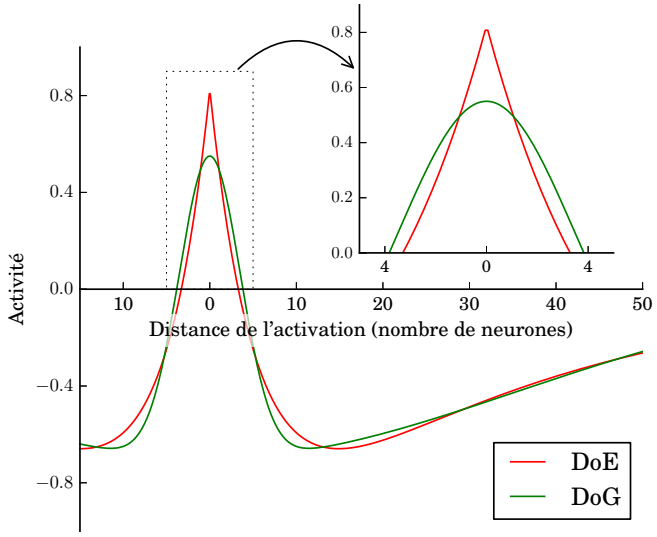


FIGURE 3.6 – Alignement de la DoE avec la DoG avec une discrétisation de 100 neurones.

TABLE 3.2 – Paramètres et leur intervalle utilisé pour l'optimisation du modèle de DNF impulsionnel avec noyau exponentiel

Paramètre	Intervalle Compétition	Compétition	Intervalle Mémoire	Mémoire de travail
iExc	$[0, 5]$	0.460	$[0, 5]$	0.789
wExc	$[0, 1]$	0.111	$[0, 10^{-3}]$	1.118×10^{-5}
iInh	$[0, 5]$	0.411	$[0, 5]$	0.589
wInh	$[0, 1]$	0.423	$[0, 10^{-3}]$	8.947×10^{-5}
τ	$[0, 1]$	0.124	$[0, 1]$	0.141
h	$[-1, 1]$	0.0	$[-1, 1]$	-0.007

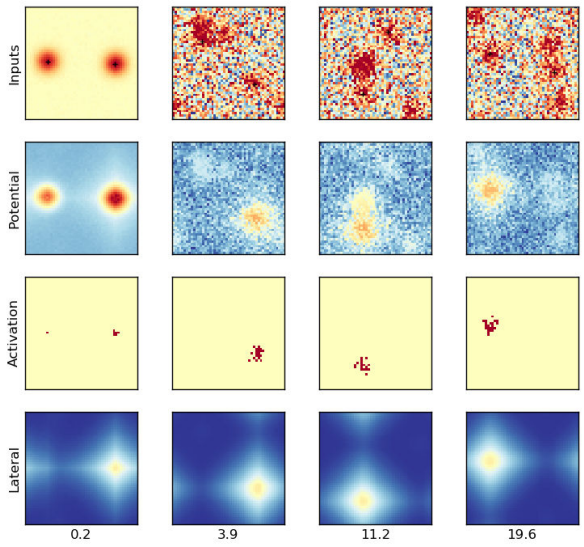
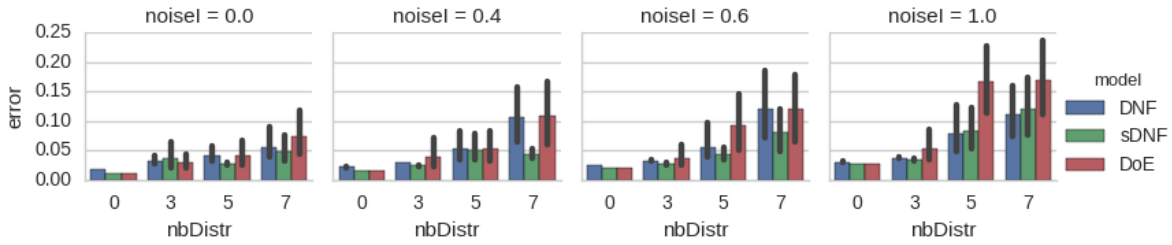
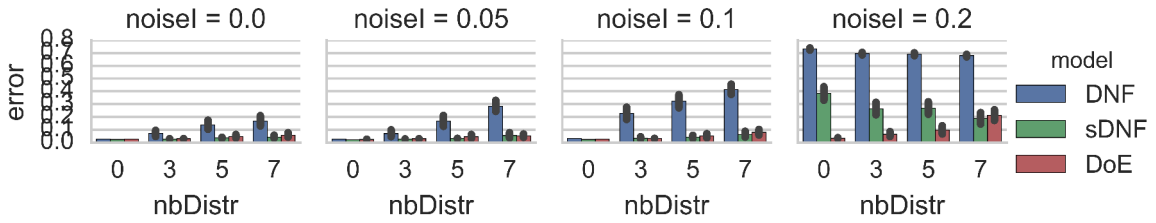


FIGURE 3.7 – Exemple de dynamique pour un DNF impulsionnel avec un noyau d'interactions latérales de type différence d'exponentielles.

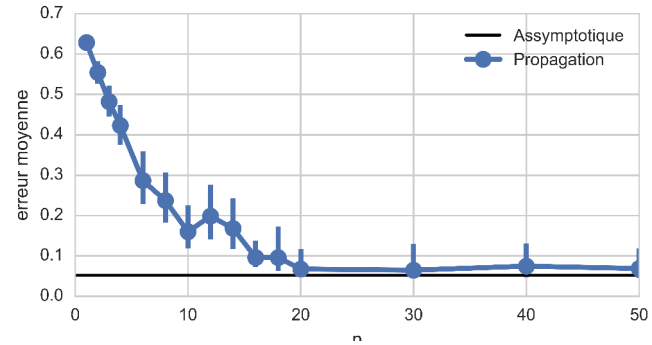


(a) Suivi

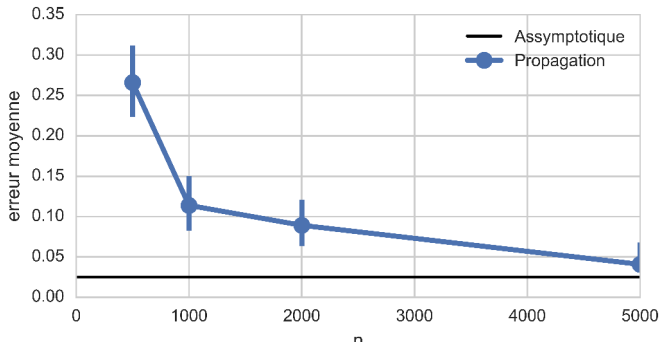


(b) Mémoire

FIGURE 3.8 – Comparaison du noyau différence d'exponentielles (DoE) par rapport au noyau standard pour le modèle impulsionnel (sDNF) et standard (DNF). En haut le scénario de suivi robuste de cible et en bas le scénario de mémoire de travail.



(a) Suivi de cible



(b) Mémoire de travail

FIGURE 3.9 – Erreur de distance en fonction de la subdivision des impulsions N. En haut, le scénario suivi de cible robuste nécessite beaucoup moins de répétitions pour tendre vers le résultat asymptotique (ligne noire) que le scénario mémoire de travail (bas).

3.4 Conditions limites

La transmission aléatoire des qImpulsions correspond à une succession de lois binomiales. On peut prouver que la distribution conditionnelle de $Y \sim B(X, q)$ conditionnelle de $X \sim B(n, p)$ est $Y \sim B(n, pq)$. Par conséquent la variable aléatoire Q_a^d correspondant au nombre de qImpulsions excitatrices reçues par un neurone à distance d du neurone activé a une distribution binomiale :

$$Q_a^d \sim B(N, p_a^d) \tag{3.17}$$

Nous souhaitons aussi déterminer le nombre d'impulsions optimal N pour assurer que la variation des résultats soit contrôlable. Nous utilisons pour cela le simulateur logiciel au niveau approché. Sur la figure 3.9 on peut voir les résultats de simulation.

Les résultats montrent que le scénario mémoire de travail est beaucoup plus sensible aux perturbations dans la transmission des informations latérales puisqu'il faut plus de 2000 répétitions pour arriver à stabiliser le comportement.

3.5 Implantation matérielle

Pour s'assurer que le modèle de DNF impulsionnel et le modèle RSDNF sont équivalents, il est nécessaire que la propagation des informations soit terminée lorsque la mise à jour des potentiels est calculée. Nous avons proposé un système de tampons permettant d'ordonnancer le routage des impulsions de manière rapide et économe. La génération des nombres aléatoires est contraignante. Il est nécessaire de générer huit nombres aléatoires par cycle de propagation et par neurone. Quatre pour les connections excitatrices et quatre pour les connections inhibitrices aux quatre voisins. Nous avons donc utilisé un générateur cellulaire de nombres aléatoires qui a l'avantage de générer un nombre important de nombres pseudo-aléatoires de bonne qualité avec une surface minimale sur FPGA [Girau and Vlassopoulos, 2011].

3.5.1 Transmission des impulsions

Illustrons d'abord les problèmes liés à la transmission des impulsions telle que nous l'avons décrite. Si un neurone reçoit une qImpulsion de chacun de ses voisins, il faut transmettre 3 qImpulsions vers le nord et le sud et une qImpulsion vers l'est et l'ouest (comme cela à été décrit plus haut). Les routeurs sont chargés des qImpulsions dirigées vers chaque direction. Ils ne peuvent transmettre qu'une seule qImpulsion à la fois. Par conséquent 2 qImpulsions doivent être stockées dans les routeurs nord et sud afin de différer la propagation des 4 qImpulsions restantes. Il arrive donc fréquemment qu'il y ait une saturation du système de routage et cela nécessite de prévoir une taille suffisante pour les tampons des routeurs et de laisser un temps de propagation assez long afin de permettre à toutes les qImpulsions de finir leur chemin.

Comme on peut le voir dans la figure 3.10, la saturation des routeurs dépend du nombre de neurones prédécesseurs activés. Pour les routeurs est et ouest proches du cluster, le nombre de neurones actifs prédécesseurs est l'ensemble des neurones actifs. Expérimentalement il est possible de déterminer le nombre de neurones activés simultanément maximal P. Sur une carte de 50×50 ($R = 50$) neurones et lors d'un scénario de suivi de cible, $P = 23$. Le nombre de qImpulsions maximal devant traverser un routeur (en considérant une probabilité $p = 1$ pour chaque transmission) sera alors de NP. Vu qu'il faut laisser le temps aux dernières impulsions de traverser la carte de neurones nous ajouterons $2R$ afin d'avoir le temps de diffusion des qImpulsions $td = NP + 2R$. Il faut aussi choisir la capacité des tampons de façon à ce qu'ils puissent stocker suffisamment de qImpulsions. Pour simplifier on considère que le temps de transmission jusqu'au routeur le plus chargé est nul. Quand la diffusion commence à $t = 0$ le tampon est déjà chargé de N qImpulsions vu que le neurone dont il dépend est actif. Il a au total $P - 1$ prédécesseurs actifs qui envoient N qImpulsions avec un débit de 3 bits par itérations. En même temps le routeur en question se décharge de 1 bit par itération. La charge maximale du tampon sera donc de $N + 2N(P - 1)$. Nous avons ainsi choisi une taille de tampon de 8 bits dans notre implémentation car c'est suffisant pour une carte avec $N = 20$ et $R = 50$.

L'asymétrie du routage provoque donc la surcharge des tampons horizontaux ce qui augmente le nombre d'itérations nécessaires la diffusion des qImpulsions. L'autre conséquence de cette surcharge est une asymétrie dans la vitesse de diffusion des qImpulsions en fonction de la probabilité de transmission de la couche de diffusion. Si la probabilité de transmission est faible, il y a moins de qImpulsions à stocker que si la probabilité de transmission est élevée. Par conséquent la couche de routage des impulsions excitatrices est plus rapide que celle des impulsions inhibitrices. Il est donc critique de stopper la diffusion lorsque les deux couches ont fini leurs transmissions. Si la diffusion est arrêtée plus tôt, il y aura un surplus d'excitation dévastateur pour la dynamique du DNF.

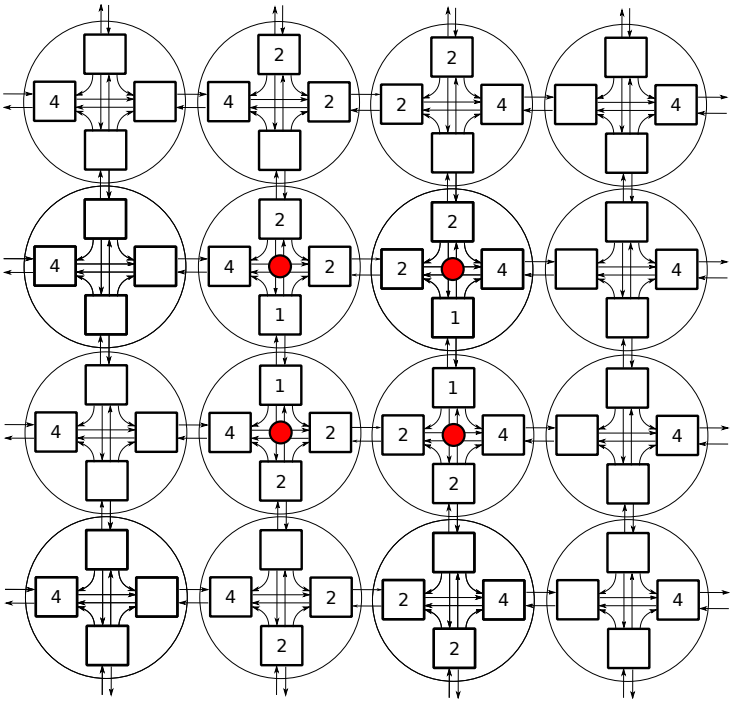


FIGURE 3.10 – Nombre de prédécesseurs de chaque routeur lorsqu'il y a un cluster d'activation de 4 neurones (en rouge).

3.5.2 Générateur de nombres aléatoires

Nous utilisons les CAPRNG pour générer les nombres aléatoires (cellular automata pseudo random numbers generators). C'est un outil déjà étudié et appliqué [Vlassopoulos and Girau, 2014] qui est pratique pour notre tâche vu que leur structure est déjà cellulaire et décentralisée.

3.5.3 Module d'entrées-sorties

Afin de valider le comportement de la version matérielle du RSDNF nous avons implanté un module d'entrées-sorties permettant de tester le comportement de la carte FPGA. Le but de ce module est de lire le fichier des entrées et d'écrire le fichier des sorties à chaque itération de calcul. Pour chacune des R lignes de neurones, les entrées sont envoyées de façon séquentielle à une ligne à retard (tapped delay line) avec une horloge R fois plus rapide que l'horloge des calculs neuronaux. La taille de chacune des R lignes à retard est égale à la largeur de la carte R. Une fois qu'une ligne est pleine, les entrées sont propagées aux neurones correspondants. Après la mise à jour des neurones, la valeur de leur potentiel est propagée dans un registre à décalage parallèle de sortie de façon à stocker ensuite de façon séquentielle les potentiels dans un fichier.

3.5.4 Résultats d'implantation sur FPGA

Une implantation VHDL a été faite de tous les modules nécessaires au fonctionnement du RSDNF. Le logiciel Xilinx 14.2 a permis la synthèse, le placement et le routage du modèle matériel sur une carte Virtex-6 xc6vlx760t-3ff1156. Les résultats de l'implantation matérielle sont présenté sur la figure 3.11. La surface grandit assez rapidement avec le nombre de neurones.

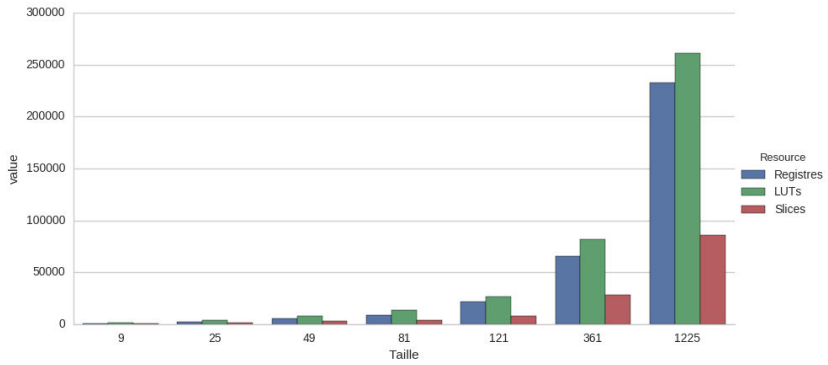


FIGURE 3.11 – Surface d'implantation pour le modèle RSDNF, avec $R = 3, 5, 7, 9, 11, 19$ et 35.

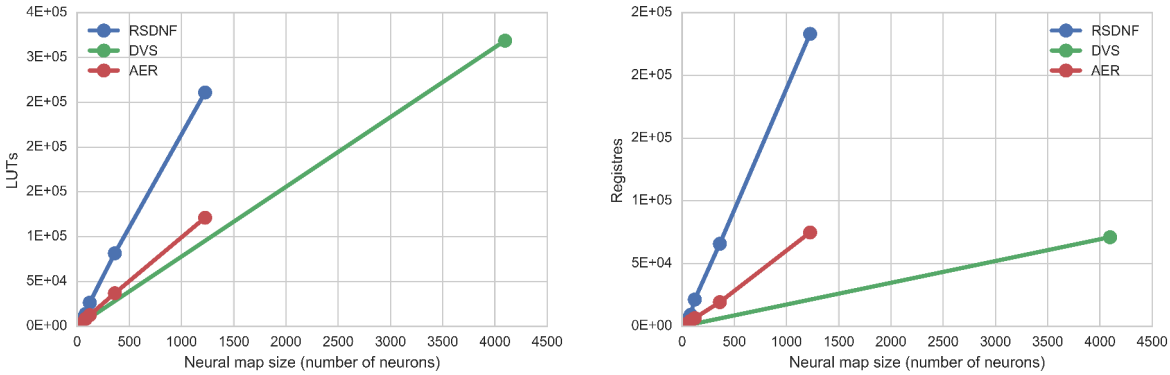


FIGURE 3.12 – Résultats d'implantation du modèle RSDNF comparé aux deux architectures centralisées basées sur le protocole AER. Pour rappel, l'architecture AER est complètement parallèle puisque chaque neurone et synapse possède des ressources dédiées. L'architecture DVS est similaire à AER mais multiplexe la mise à jour des neurones.

On peut comparer la mise à l'échelle de l'architecture aux implantations centralisées que nous avons introduites au chapitre précédent basées sur le protocole AER (fig 3.12).

On voit alors que la mise à l'échelle du modèle matériel RSDNF est peu compétitive par rapport à des implantations plus conventionnelles. Le nombre de registres notamment est beaucoup plus important que pour le modèle AER. Ainsi le gain en mise à l'échelle dû à la structure cellulaire est pénalisé par la surface d'implantation.

3.6 Optimisation de l'implantation matérielle

Dans un processus itératif d'amélioration des performances computationnelles et matérielles du modèle, nous avons vu que la robustesse du modèle DNF permettait la facilitation de son implantation matérielle et que les modifications apportées permettent parfois une amélioration de certaines caractéristiques comportementales. Nous allons continuer dans cette direction en cherchant à optimiser un aspect précis de l'implantation matérielle : la génération de nombres

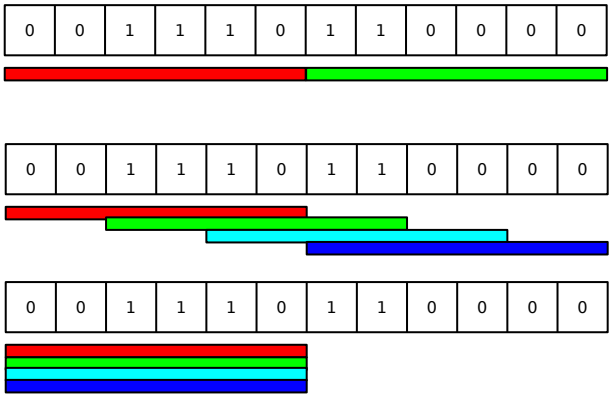


FIGURE 3.13 – Partage des nombres pseudo-aléatoires (PRN en blanc) par les routeurs (en couleur) d'un neurone. Dans le premier cas chaque routeur utilise des PRN différents le décalage est égal à la précision du nombre aléatoire du routeur (ici 6). Dans le deuxième cas il y a un décalage de 2 unités et dans le dernier cas, tous les routeurs d'un neurone utilisent le même PRN : décalage de 0.

aléatoires.

En effet, les résultats d'implantation montrent qu'un tiers de la surface est utilisée par les générateurs de nombres aléatoires. Ceci limite beaucoup la mise à l'échelle de ce modèle et explique en partie le fait que l'on ait été capable d'implanter un champ neuronal de résolution 35×35 uniquement sur un FPGA de type Xilinx xc6vlx760t-3ff1156.

Il y a de nombreuses façons d'améliorer la surface dédiée aux nombres aléatoires et nous avons choisi d'explorer deux pistes. La première piste est de partager les ressources aléatoires de façon à optimiser leur utilisation, la deuxième est de pré-calculer des bits aléatoires et de les diffuser dans le réseau de façon à éviter les corrélations.

3.6.1 Partage des ressources aléatoires

L'idée principale est de générer un nombre aléatoire de P_n bits par neurone et d'obtenir les P_r bits nécessaires a chacun des 8 routeurs par un décalage parmi les P_n bits. Par exemple la figure 3.13 montre le cas où 4 routeurs partagent $P_n = 24$ bits (seul 12 sont montrés). Dans l'exemple de la figure $P_r = 6$.

Ces expériences montrent qu'en utilisant un décalage $S_r = 0$, il n'y a pas de différence significative avec l'implantation. C'est donc une très bonne nouvelle puisqu'on peut d'ores et déjà diviser la surface dédiée aux générateurs aléatoires par 8.

Si on se base sur une précision $P_r = 8$ pour les lois de Bernoulli, on a besoin d'un seul CAPRNG de 8×8 cellules pour 8 neurones. Ce qui donne pour une résolution $R = 35$ un total de 154 CAPRNG pour une surface d'implantation (sans optimisation particulière) de 9984 bascules et 7804 LUT.

3.6.2 Pré-génération des nombres aléatoires

Potentiellement, la pré-génération de nombres aléatoires pourrait être beaucoup plus économique car elle ne nécessite aucun calcul. Elle nécessite uniquement une bascule (sans reset) par routeur ce qui fait un total de $35 * 35 * 8 = 9800$ bascules pour une résolution $R = 35$. Le

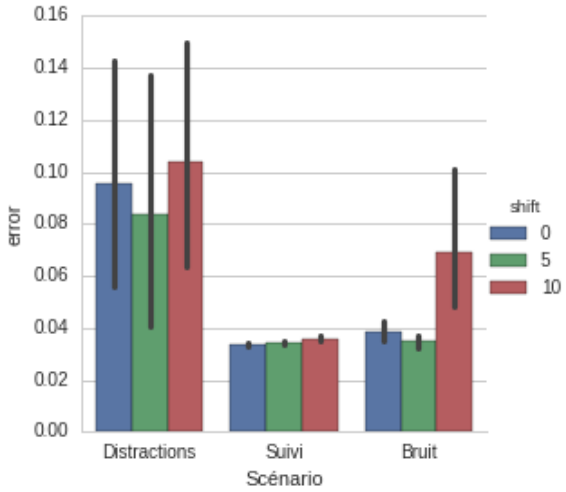


FIGURE 3.14 – Erreur de distance du modèle RSDNF avec différents degrés de partage des ressources aléatoires au sein du même neurone.

nombre de bascules est donc le même que la version CAPRNG partagés, mais aucune LUT n'est nécessaire.

Cependant la conception est un peu plus complexe vu que l'on peut très facilement introduire de nombreuses corrélations qui biaisent l'approximation de la différence d'exponentielles qu'on calcule lors de la propagation aléatoire.

3.6.2.1 Corrélation 1D

Pour illustrer les problèmes liés à la corrélation des nombres aléatoires nous allons faire quelques expériences avec la propagation d'une seule impulsion subdivisée en N qImpulsions dans une seule direction d'un tableau unidimensionnel.

A $t = 0$ des bits aléatoires sont pré-calculés avec une probabilité $p = 0.93$ et sont chargés dans les routeurs. Le neurone le plus à gauche est activé et transmet ses N qImpulsions vers la droite. Les qImpulsions sont transmises de routeur en routeur toujours vers la droite si le bit aléatoire stocké dans le routeur est haut.

On peut voir sur la figure 3.15 l'effet que peut avoir la corrélation des nombres aléatoires sur la propagation des qImpulsions dans deux cas. Dans le premier cas, les bits aléatoires sont transmis de proche en proche vers la droite (même sens de propagation que les qImpulsions) ou vers la gauche (sens inverse des qImpulsions). La propagation des qImpulsions dure $N + R$ itérations.

Il apparaît que la propagation des nombres aléatoires dans le même sens ou dans le sens opposé de la propagation des impulsions donne des résultats plus ou moins biaisés. Dans la version 2D on évitera donc ce type de propagation.

Même si la propagation ne se fait pas dans le même sens, il y aura quand même une période de propagation. La période de propagation est le temps pour qu'un nombre aléatoire revienne dans le même routeur. Nous allons maintenant étudier l'influence de cette période sur la dégradation de la qualité de la propagation synaptique. Si la propagation se fait en sens inverse sur la totalité

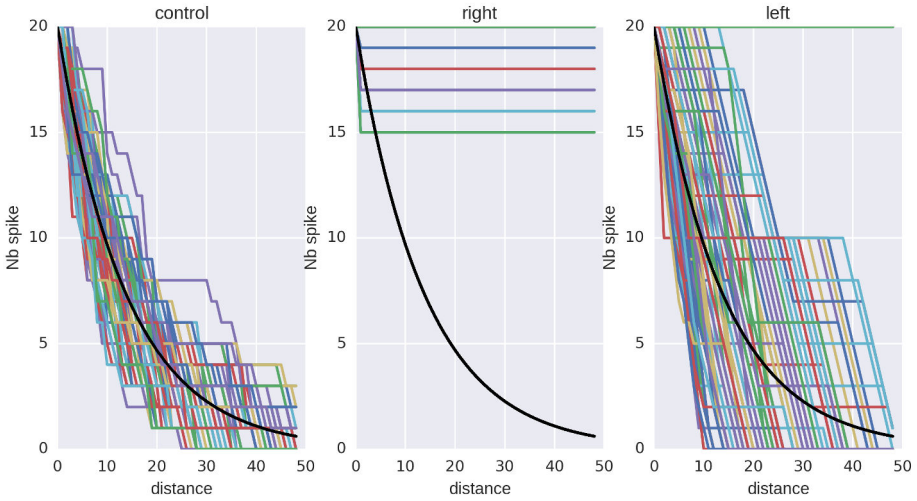


FIGURE 3.15 – Propagation des qImpulsions dans les différent modèles de génération aléatoire. La propagation est effectuée avec $N = 20$. L'évolution du nombre de qImpulsions reçues en fonction de la distance à la source est indiquée pour différentes exécutions. A gauche propagation des qImpulsions avec un bit aléatoire généré pour chaque transmission. Au centre, les bits sont précalculé et propagés dans la même direction que les qImpulsions (vers la droite). A droite, les bits sont propagés dans la direction opposée (vers la gauche). La courbe noire montre la quantité asymptotique de qImpulsions à distance donnée.

de la ligne, la période est R. On peut simuler des périodes inférieures pour voir l'influence de la période sur la propagation d'un paquet de N qImpulsions.

Pour quantifier la différence entre une propagation aléatoire et une propagation parfaite, nous utilisons la racine de l'erreur quadratique normalisée (NRMSE) calculée entre la diffusion obtenue avec une couche de diffusion du modèle aléatoire $\bar{D}$ et une diffusion parfaite D :

$$NRMSE = \frac{\sqrt{\frac{\sum_{i=1}^{R}\sum_{j=1}^{R}(\bar{D_{i,j}}-D_{i,j})^2}{R^2}}}{max(D) - min(D)}. \tag{3.18}$$

On voit sur la figure 3.16 que la période influence la moyenne de l'erreur de manière exponentielle. Et l'influence devient nulle à partir du moment ou elle est plus grande que N. On essaiera donc de maximiser la période dans la version 2D.

3.6.2.2 Corrélation 2D

La propagation sur la grille bi-dimensionnelle est définie de façon ad hoc en se basant sur les conclusions du paragraphe précédent. Sur la figure 3.17a on peut voir que le sens de propagation des bits aléatoires n'est ni dans la même direction que les impulsions ni dans la direction opposée mais dans la direction perpendiculaire. De cette façon on évite les problèmes de corrélation décrits dans le paragraphe précédent (voir figure 3.15).

De façon à varier les types de période on introduit deux types de routage aux bords. Le premier définit une propagation courte car il connecte la fin d'une ligne (resp. colonne) au début de la même ligne (resp. colonne). La période de propagation est donc de R (voir figure 3.17b).

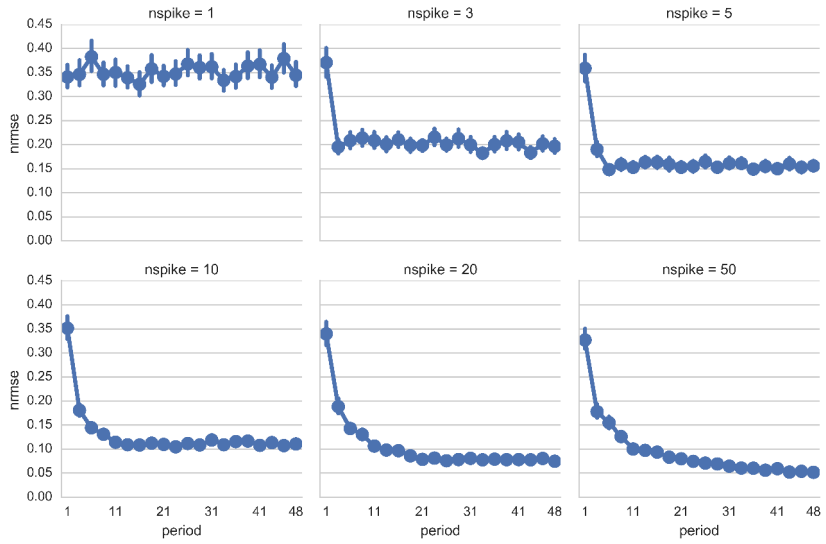


FIGURE 3.16 – Propagation des qImpulsions avec différentes périodes de propagation. La propagation est effectuée avec N compris entre 1 et 50. L'erreur normalisée du nombre de qImpulsions reçues est indiquée en fonction de la période introduite dans la propagation (en sens inverse) des nombres aléatoires.

Le deuxième type de connexion aux bords permet une propagation plus longue car la fin d'une ligne (resp. colonne) est connectée au début d'une autre ligne (resp. colone) qui propage les bits dans la même direction. Dans ce cas la période sera de $R * (R/2)$ (voir figure 3.17c).

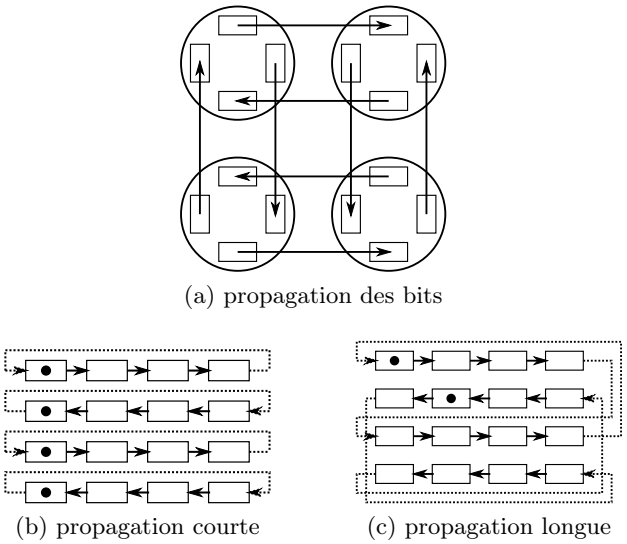
Pour enlever le biais de la période on propose l'étude d'une variation de chaque type de propagation. On propose ainsi d'ajouter un générateur aléatoire sur chaque cycle de façon à ne jamais avoir un bit aléatoire identique deux fois dans le même routeur. On appellera cette version la version mixte. Il y a $8R$ PRNG dans la version de propagation courte mixte et 16 dans la version de propagation longue mixte.

On teste ici la propagation des qImpulsions dans le cas où $P = 10$ neurones sont activés au centre d'une carte de résolution $R = 49$. La NRMSE est calculée en comparant avec une diffusion parfaite D définie comme la convolution de l'activité $\mathcal{A}$ des neurones avec la fonction des poids latéraux W (une DoE dans ce cas) :

$$D_{x,y} = \sum_i \sum_j \mathcal{A}_{i,j} W_{x-i,y-j}. \tag{3.19}$$

Les résultats montrent plusieurs choses intéressantes (voir figure 3.18). Les architectures à propagation de bits pré-calculés ont des résultats plus mitigés que le routage de type CAPRNG partagé. La pire est l'architecture à période courte puisque sa petite période de 49 itérations est beaucoup plus petite que le temps total de propagation avec 20 qImpulsions. La propagation à longue période a quant à elle une période de 1200 itérations ce qui est bien plus grand que les 298 itérations accordées à la propagation avec 20 qImpulsions.

Mais au delà de l'erreur constatée à chaque itération, il est intéressant de vérifier qu'il n'y a pas de biais à long terme. En effet les nombres aléatoires ne sont pas recalculés entre chaque phase de propagation. Par conséquent il est important de vérifier que la moyenne reste proche de la moyenne espérée après les 100 répétitions de l'expérience.



(a) propagation des bits

(b) propagation courte

(c) propagation longue

FIGURE 3.17 – Schéma de propagation des bits aléatoires. a) les bits se propagent entre routeurs de même direction mais dans la direction perpendiculaire à la direction du routeur. La direction de propagation est inversée de ligne en ligne et de colonne en colonne. b) dans le mode de routage "court" la fin d'une ligne (resp. colonne) est connectée au début de la même ligne (resp. colonne). c) dans le mode de routage "long" la fin d'une ligne (resp. colonne) est connectée au début d'une autre ligne (resp. colonne) qui propage les bits dans la même direction. Les versions mixtes de ces deux types de propagation contiennent des générateurs de nombres pseudo-aléatoires représentés par des points noirs.

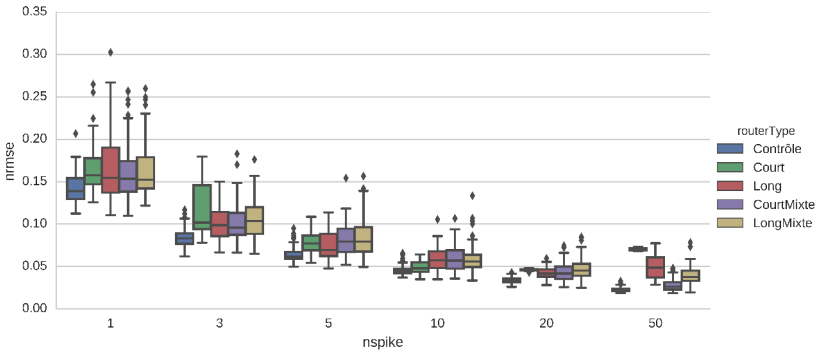


FIGURE 3.18 – Erreur quadratique moyenne normalisée d'une propagation sur une grille bi-dimensionnelle de 10 impulsions subdivisées en N qImpulsions. Les différents modèles correspondent à différentes façons de propager les nombres aléatoires.

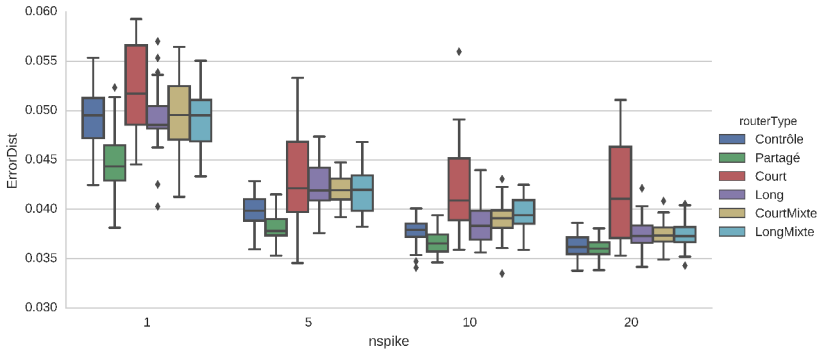


FIGURE 3.19 – Erreur de distance lors de la simulation des champs neuronaux avec différents types de générateurs aléatoires sur le scénario témoin.

3.6.3 Résultats expérimentaux

Nous avons testé les générateurs décrits dans la section précédente dans les scénarios de tests utilisés tout au long de cette thèse.

Le premier scénario testé est le scénario Témoin. Le premier test avait pour but de valider le comportement émergent du DNF et d'éliminer certains modèles qui auraient les mêmes comportements [8]. On voit effectivement sur la figure 3.19 que les routages de type long, court mixte ou long mixte ont les mêmes moyennes avec un risque du premier type $\alpha = 0.05$. On étudiera donc uniquement le routage long par la suite.

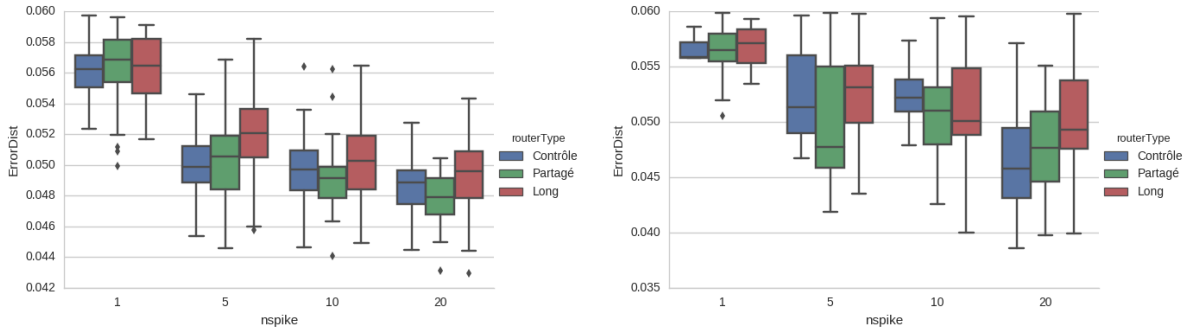
Les résultats pour les trois autres scénarios confirment la tendance générale : le partage des CAPRNG est aussi bon que le témoin et le pré-calcul des nombres aléatoires donne en moyenne des résultats moins bons que le témoin bien que la différence ne soit pas significative en enlevant les valeurs aberrantes avec un test de student apparié avec la correction de Bonferroni (voir figure 3.20).

Nous sommes parvenus grâce à cette étude à diviser la surface d'implantation des nombres aléatoires par 8 dans un premier temps (version CAPRNG partagée) puis à enlever toutes les LUT dédiées à la génération des nombres aléatoires en pré-calculant les bits aléatoires et en les propageant dans la carte. Nous avons vu que si on garde un nombre raisonnablement réduit de qImpulsions le schéma de propagation que l'on a proposé suffit pour conserver le comportement du DNF sans avoir à ajouter une source de bits aléatoires.

L'impact sur la surface totale du RSDNF est montré sur la figure 3.21. On voit que l'économie des LUT dans la version partagée n'est pas très importante par rapport à la quantité totale.

8. Pour s'assurer d'une différence significative de moyenne entre les différentes distributions on procède comme suit : 1) test de normalité de chaque distribution avec le test de Shapiro-Wilk. 2) test d'homoscédasticité (homogénéité des variances) avec le test de Barlett. Si les deux conditions sont respectées on procède alors au test d'ANOVA qui teste l'hypothèse nulle d'égalité des moyennes puis au test HSD de Tukey pour l'égalité appariée de chaque distribution.

Dans notre cas les distributions seront normales mais leur variance sera différente. Par conséquent on teste la différence de moyenne avec l'ANOVA non paramétrique de Kruskall-Wallis et on utilisera le test de Student corrigé avec la correction de Bonferroni pour évaluer la différence entre les distributions deux à deux.



(a) Bruit (b) Distracteurs

FIGURE 3.20 – Comportement du modèle RSNDF avec différents types de génération de nombres aléatoires. Pour des scénarios de suivi robuste en fonction du nombre de subdivisions des impulsions N.

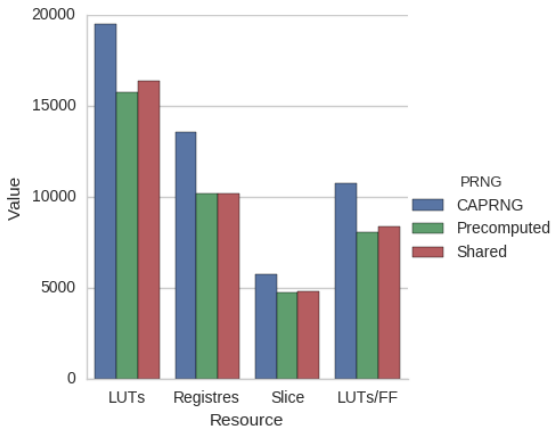


FIGURE 3.21 – Surface d'implémentation du modèle RSDNF avec différentes optimisations pour la génération des nombres aléatoires. En bleu un automate cellulaire de génération de nombres pseudo-aléatoires (CAPRNG) est utilisé en dédiant des bits aléatoires à chaque routeur. En rouge, le même automate est utilisé mais les bits sont complètement partagés au sein d'une cellule. En vert, il n'y a pas de calcul de nombres aléatoires : des bits aléatoires sont pré-calculés et propagés dans la carte.

Surtout que cette version diminue la quantité de couples LUTs-FF ce qui entraîne un gain en slices relativement faible (100 slices environ).

La robustesse des DNF a donc permis une optimisation conséquente de la surface d'implantation. La résistance au bruit que nous avons déjà montré permet ici au modèle de résister à une variance plus grande dans la communication inter-neurones. Cette variance ne change pas les résultats de manière significative surtout quand la communication est très imprécise comme c'est le cas quand le nombre de qImpulsions est faible. Cette étude pourrait être améliorée en formalisant un critère mathématique permettant de mesurer les différents biais introduits par la réutilisation des bits aléatoires et en améliorant le flux des bits aléatoires en se basant sur ce critère. Cela ne changerait pas la surface d'implantation mais pourrait améliorer le comportement en le rapprochant de celui du témoin. Cela dépasse le cadre de cette étude.

3.7 Robustesse matérielle

La robustesse est une des caractéristiques recherchées de notre architecture. Nous avons vu que les DNF sont robustes au bruit, aux distractions et aux perturbations dans la communication inter neurones.

Nous allons voir dans cette section qu'il y a un type de robustesse pour lequel notre implantation matérielle est moins satisfaisante, c'est la robustesse aux fautes matérielles non ponctuelles.

Nous allons commencer par définir les différents types de fautes matérielles avant d'étudier leur impact sur le comportement des RSDNF.

3.7.1 Les fautes matérielles

L'ITRS (International Technology Roadmap for Semiconductors) prévoit une échelle de fabrication des semi-conducteurs CMOS à 10nm en 2017 et 5nm en 2019 [ITRS, 2014]. L'augmentation de la densité et de la complexité des circuits implique énormément de tests pour assurer un fonctionnement 100% correct à la fabrication et assurer la fiabilité et la disponibilité des circuits.

Par conséquent, le gain en coût et rapidité de développement pour du matériel robuste aux fautes matérielles passagères et permanentes serait très important.

C'est un des arguments majeurs pour le développement de matériel bio-inspiré, neuromimétique ou cellulaire. C'est par exemple le cas du projet FACET qui développe du matériel neuromimétique analogique directement sur plaquette (wafer scale) grâce à une bonne tolérance aux fautes [Schemmel et al., 2010].

On distingue plusieurs types de fautes. Les événements unitaires (single event upset, SEU en anglais) sont des fautes généralement non permanentes principalement causées par des particules secondaires libérées par la collision d'un neutron avec un atome de silicium ou d'un contaminant produisant une particule alpha. Les neutrons sont créés lors du passage des rayons cosmiques et des protons de l'espace interagissent avec l'atmosphère terrestre. L'énergie d'un neutron peut atteindre 1000 MeV (méga électron volt) ce qui peut avoir des impacts variés sur les composants : rupture de porte, court-circuit, etc. Il serait possible de protéger les circuits de ce genre de faute mais les boucliers sont trop encombrants pour que ce soit pratique. Par exemple il faut 30 mètres d'eau pour arrêter les neutrons à haute énergie [Hussein and Swift, 2015].

Les événements unitaires peuvent aussi produire des fautes permanentes ainsi que des erreurs d'impression à la construction du circuit. Les erreurs à la construction sont généralement détectées par des tests et le circuit fautif sera jeté.

Bien que nous n'ayons pas encore l'intention de produire notre circuit à grande échelle ou de l'envoyer sur mars, il est intéressant d'étudier comment le substrat de calcul que nous avons

proposé se positionne par rapport à ces problématiques.

La résistance aux fautes sur FPGA est un vaste domaine de recherche et de nombreux outils existent pour tester la logique et la connectique. Les FPGA industriels possèdent nativement des mécanismes de restauration des fautes (échange des ressources fautives avec des ressources libres) et de résistance aux Single Event Effects (latch up resistance).

Nous allons étudier l'influence des fautes temporaires et des fautes permanentes sur notre architecture avec des outils simplifiés.

3.7.2 Modélisation

Nous utiliserons l'implémentation logicielle cellulaire du modèle pour modéliser les SEU. Cela ne nous permettra pas de modéliser tous les types de fautes mais un nombre suffisant pour avoir une idée de l'impact que peuvent avoir des fautes temporaires ou permanentes sur le comportement du modèle.

Pour comprendre comment nous avons modélisé les erreurs il est nécessaire de comprendre comment est conçu le simulateur matériel.

La classe *Module* possède une liste de paramètres constants lors de la simulation, une liste de sous-modules, une liste de registres et une liste de voisins. La mise à jour des registres est effectuée dans une méthode *compute* redéfinie pour chaque type de module.

Lorsque toutes les méthodes *compute* de tous les modules ont été exécutées, une méthode de synchronisation *synch* est appelée sur tous les modules. L'appel de cette fonction simule une horloge de synchronisation globale qui permet aux registres de prendre leur nouvelle valeur.

Les SEU sont modélisés au niveau de chaque registre avec un *masque d'erreur* de la taille du registre de cette manière :

$$val \leftarrow nextval \oplus errormask$$

Ou *errormask* est le masque d'erreurs. Un SEU sera donc une inversion de bit dans un des registres de tout le modèle. Ceci correspond à une erreur temporaire.

Les erreurs permanentes seront modélisées en forçant un certain bit du registre à 1 ou 0 avec un masque similaire.

Ce modèle est incomplet car il ne permet pas de modéliser un SEU dans une LUT ou dans la connectique. La simulation de telles erreurs aurait un niveau de complexité bien plus élevé.

3.7.3 Erreur permanente sur une couche de routage

Dans un premier temps nous allons parcourir de façon exhaustive tous les bits de tous les registres afin de les forcer chacun à son tour à «1» ou «0» et de voir l'effet que cela a sur une phase de routage des impulsions. La carte contient 49×49 unités, la probabilité de transmission est de 0.93 et il y a 10 neurones activés au milieu avec 20 subdivisions par impulsions. Comme dans la section précédente, on compare le résultat obtenu à une propagation parfaite en utilisant l'erreur quadratique moyenne normalisée.

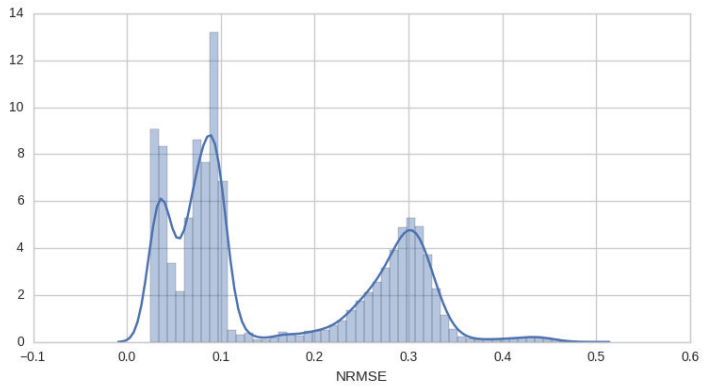
Les registres concernés ainsi que leur taille sont indiqués dans le tableau 3.3.

Au total il y a 2401 modules «CellRsdnf» et chacun a 4 routeurs, ce qui donne un total de 132055 bits à tester sur une couche de routage.

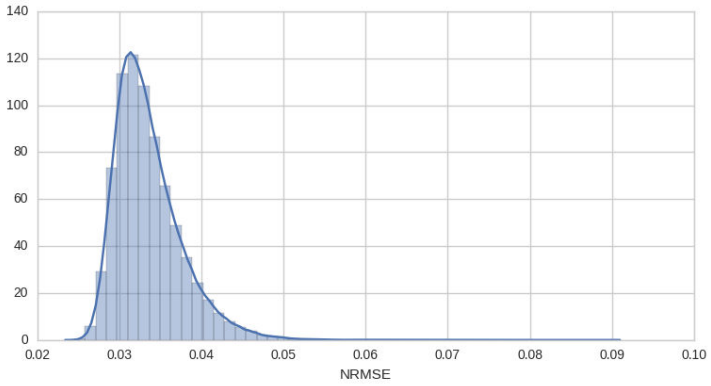
La figure 3.22 montre que les erreurs permanentes où les bits sont forcés à «1» ont un impact beaucoup plus important sur la propagation des impulsions. Cet impact suit une distribution tri-modale. L'erreur entre 0 et 0.05 est insignifiante puisqu'elle est due à la propagation aléatoire des impulsions. On distinguera donc les erreurs légères ($e \in [0.05, 015]$) et graves ($e > 0.15$). Les erreurs les plus graves se situent autour de $e = 0.3$ et correspondent à un registre d'activation

TABLE 3.3 – Registres associés à chaque module avec leur taille.

Module	Registre	Taille	Description
CellRsdnf	activated	1	signale une activation du neurone
CellRsdnf	nb bits received	10	nombre de qImpulsions reçues lors d'une phase de propagation
Router	buffer	10	tampon de stockage des qImpulsions
Router	spike out	1	qImpulsion émise par le routeur



(a) Bit forcé à «1»



(b) Bit forcé à «0»

FIGURE 3.22 – Erreur permanente sur une phase de propagation. En haut, les bits sont forcés à «1» les uns après les autres et en bas ils sont forcés à «0». L'histogramme des erreurs observées est tracé puis approché par une courbe.

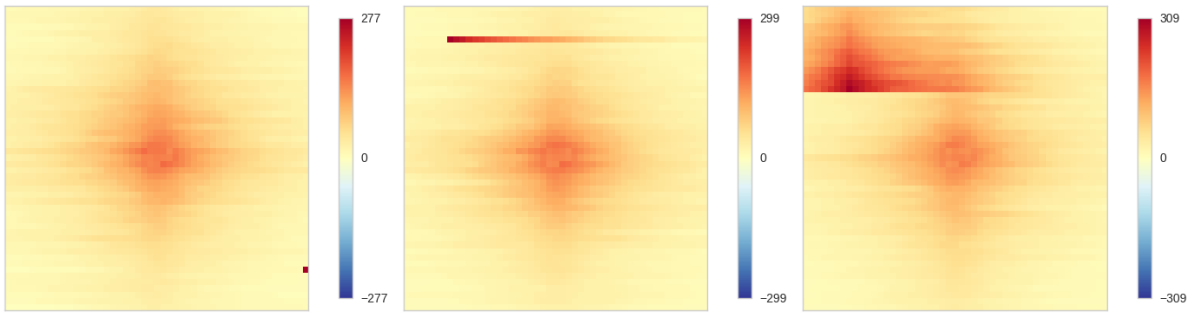


FIGURE 3.23 – Exemple d'erreur permanente sur une phase de propagation. Les deux premières erreurs sont légères puisque localisées. La dernière est plus grave.

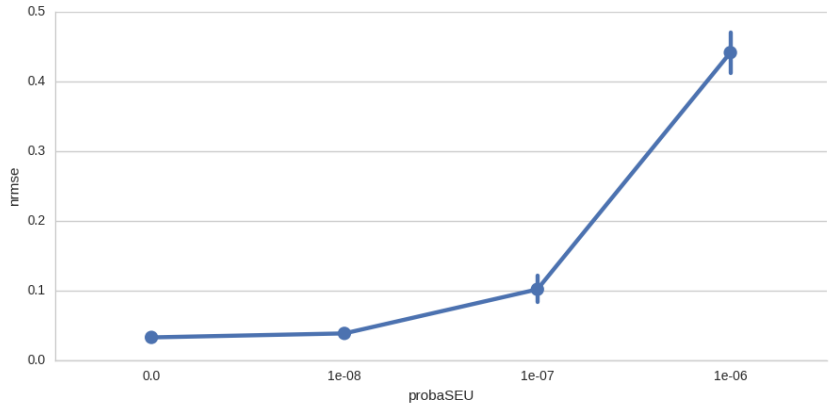


FIGURE 3.24 – Erreur de propagation avec une probabilité d'erreur temporaire croissante.

forcé à «1» ou un tampon vertical toujours actif . Un erreur légère correspond à un tampon horizontal toujours actif (voir figure 3.23).

3.7.4 Erreur temporaire sur une couche de routage

Les erreurs temporaires sont calculées en appliquant une seule fois le masque d'erreur. Étant donné qu'il est beaucoup plus coûteux d'être exhaustif on se contentera de simuler la propagation des impulsions de la même façon mais avec un pourcentage d'erreurs temporaires croissant. On suppose donc qu'à chaque pas d'horloge, chaque bit à une probabilité p de s'inverser. Si la probabilité est grande il est donc possible que plusieurs bits s'inversent pendant une phase de propagation.

Sur la figure 3.24 on voit que l'erreur augmente rapidement avec la probabilité de SEU.

3.7.5 Conséquences sur le comportement

On voit donc que les fautes permanentes imposant de fortes valeurs dans les registres ont un impact important sur les valeurs latérales dans la moitié des cas. Mais quel sera l'impact sur le comportement du DNF ?

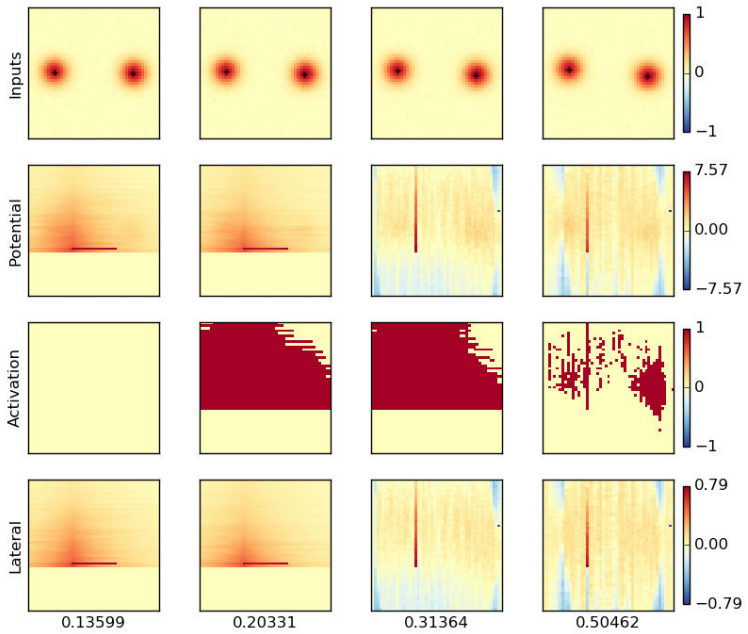


FIGURE 3.25 – Conséquences d'une erreur permanente sur la couche de routage.

Des simulations sur les champs neuronaux dynamiques montrent une certaine résistance du modèle aux fautes. Que ce soit au niveau du potentiel ou de l'activation, des fortes valeurs imposées de façon permanente n'ont pas d'impact si un ou deux neurones sont affectés. Pour les faibles valeurs (par exemple si l'activation est toujours nulle) alors la tolérance est encore plus grande. Ces résultats montrent que l'on peut accepter un neurone fautif à la fabrication ou une faute temporaire sur un neurone pendant l'exécution. C'est un seuil de résistance aux fautes qui serait déjà très intéressant pour une utilisation de l'implémentation matérielle en condition normale où il est très rare que plus d'un neurone soit fautif.

Malheureusement les conséquences sont beaucoup plus graves si l'on considère le modèle matériel RSDNF. En effet un routeur qui est activé en permanence finit par activer les neurones les plus proches et crée donc une activation en cascade de beaucoup de neurones perturbant irrémédiablement la dynamique du RSDNF.

On peut voir par exemple sur la figure 3.25 le cas où il y a une erreur sur la couche de routage excitatrice et une sur la couche inhibitrice.

Malgré la robustesse neuronale des champs neuronaux dynamiques, cette première étude non-exhaustive sur la tolérance aux fautes matérielles de l'architecture RSDNF révèle donc des résultats mitigés. Les raisons de la sensibilité aux fautes sont multiples. La première est le nombre important de registres à plusieurs bits qui sont très sensibles aux SEU dans les bits de poids fort puisqu'ils peuvent engendrer des valeurs élevées qui perturbent immédiatement les résultats.

Un autre facteur est le routage asymétrique dont nous avons déjà parlé et qui rend la propagation des impulsions sensible à un nombre trop élevé de qImpulsions à router. Si la phase de

propagation n'est pas assez longue, il y aura un déséquilibre massif entre la couche excitatrice et la couche inhibitrice qui déstabilisera encore plus le modèle.

Il est de plus à noter que tout modèle synchrone est sensible aux pannes d'horloge et cela est le cas de notre architecture qui n'est pas vraiment décentralisée puisqu'elle possède une horloge centrale. Néanmoins cette horloge commune n'est pas inhérente au modèle qui pourrait utiliser une horloge différente pour chaque neurone et un mécanisme de synchronisation globale pour démarrer un nouveau cycle de propagation.

* * *

Les champs neuronaux dynamiques impulsionnels aléatoires permettent une implémentation matérielle du modèle de champs neuronaux continus répondant à plusieurs contraintes. La première contrainte est un objectif de distribution et parallélisme massif, inspiré du calcul cellulaire où la mémoire et les calculs sont locaux afin d'explorer des solutions aux limites des architectures de von Neumann. Cette approche est contraire aux approches neuromimétiques classiques puisqu'elle n'utilise pas le protocole AER établi depuis longtemps comme standard de communication matérielle entre différents modules. Elle se place dans une approche cellulaire qui ne souffre pas a priori de la limite en bande passante d'un bus de communication. La communication entre les modules se veut topographique et est tout à fait cohérente avec des applications à base de rétines artificielles ou des caméras intelligentes puisque les cellules de la caméra pourraient être directement reliées aux neurones du DNF.

En pratique cependant, un goulot d'étranglement apparaît aussi dans cette implantation cellulaire des DNF : le temps de propagation des informations vu que la communication doit être globale dans certaines applications.

On peut cependant penser que la mise à l'échelle de la *diffusion* d'information devrait être plus facile qu'une communication basée sur un bus de données. Le temps de diffusion dans une carte bi-dimensionnelle est en racine carée du nombre de neurones alors qu'il est linéaire dans le cas d'un bus.

Pour l'implantation que nous avons présentée dans ce chapitre, ce n'est cependant pas le cas : le temps de propagation augmente énormément à cause de la subdivision des impulsions et du routage XY asymétrique et cela limite forcément la mise à l'échelle.

De plus la tolérance aux fautes matérielles, aspect attrayant du calcul cellulaire, n'est pas assurée et ce, malgré la robustesse au bruit inhérente aux DNF.

Cette première implémentation a cependant permis de mettre en place un cadre de développement et de test et a permis de mettre en valeur les points critiques d'une implantation cellulaire.

Chapitre 4

Grille cellulaire de DNF impulsionnels aléatoires et asynchrones

Le modèle matériel CASAS-DNF (pour Cellular Array of Stochastic Asynchronous Spiking DNF [Chappet de Vangel et al., 2015b]) est une réponse aux problématiques posées par le modèle RSDNF et plus particulièrement au problème de diffusion asymétrique des impulsions qui augmente énormément le temps de diffusion et qui impose un signal de synchronisation global pour arrêter la diffusion et passer à l'itération suivante et décuple les conséquences de certaines fautes matérielles. La diffusion asymétrique est liée au mode de routage choisi pour la diffusion des paquets, en l'occurrence les qImpulsions représentant des quanta d'impulsion codés sur un seul bit. Les tampons nécessaires à la diffusion de ces impulsions provoquent un retard de la diffusion vers les directions est et ouest proportionnel au nombre de qImpulsions à diffuser.

Le but principal du modèle CASAS est donc de se passer de la mise en tampon des qImpulsions afin d'éviter ces retards et ainsi d'éviter les problèmes liés à celui-ci.

L'architecture proposée ici se base sur l'arithmétique à base de flux de bits, que l'on a présentée dans le second chapitre de ce manuscrit.

La progression et la méthodologie suit le même ordre que le chapitre précédent et utilise les mêmes outils. En revanche, ce chapitre sera conclu par une argumentation quant à la différence entre l'architecture RSDNF et l'architecture CASAS-DNF introduite ici.

4.1 Description

De la même façon que dans RSDNF, la fonction des poids latéraux est calculée à partir d'une propagation d'informations dans la carte avec une probabilité de réception décroissante de façon exponentielle. Nous allons cependant nous placer dans le contexte des réseaux de neurones à arithmétique stochastique. Lorsqu'un neurone est activé il génère une impulsion et cette impulsion sera transmise de proche en proche, traversant à chaque fois une synapse.

L'impulsion est représentée par une valeur d'activation égale à 1 et cette valeur est multipliée par un poids synaptique à chaque fois qu'elle traverse une synapse. Les neurones peuvent alors intégrer directement l'information latérale à chaque réception.

Dans le cadre de l'arithmétique stochastique, l'impulsion sera encodée par un flux de bits de taille N et la multiplication sera réalisée avec une porte ET connectée à un flux de bits de valeur correspondante au poids synaptique. Pour l'instant nous avons utilisé un formalisme différent pour décrire ce que fait déjà le modèle RSDNF : l'impulsion subdivisée en N qImpulsions correspond à l'encodage de la valeur 1 dans un flux de bits de taille N et la propagation des

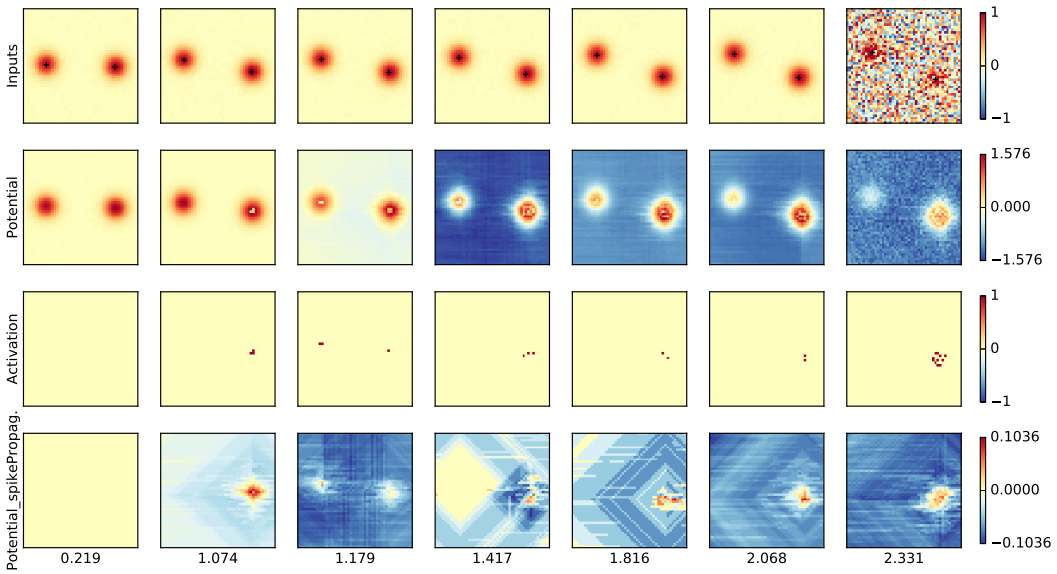


FIGURE 4.1 – Premières itérations de la simulation cellulaire du modèle matériel CASAS. Les influences latérales (dernière ligne) sont propagées à l'aide de flux de bits.

qImpulsions en fonction d'une expérience de Bernoulli de probabilité p correspond à la multiplication successive du flux de bits impulsionnels par des flux de bits constants de valeur p. Un flux de bits est donc une estimation d'une distribution de Bernoulli et la porte ET estime le résultat de la conjonction de deux distributions.

La différence se situe au niveau de l'addition des informations. Dans le modèle RSDNF, les qImpulsions sont accumulées dans des tampons de façon à ce qu'elles soient additionnées une par une dans les accumulateurs des neurones avant d'être intégrées dans le potentiel membranaire du neurone. Ici les flux de bits sont additionnés en utilisant la porte OU ce qui permet une propagation des informations *sans stockage*.

Cependant l'addition de flux de bits ne peut pas se faire sans perte. Les deux moyens disponibles pour additionner des flux de bits sont la porte OU et le multiplexeur (MUX). L'addition des flux de bits A et B en utilisant une porte OU donne en sortie $A + B - AB$. Avec un multiplexeur, on obtient $(A + B)/2$. La porte OU dégrade moins l'addition que le multiplexeur (voir figure 4.2), on l'utilisera donc pour remplacer toutes les additions.

Nous commencerons par étudier et définir le modèle afin qu'il soit le plus proche possible du modèle DNF tel que nous l'avons défini dans l'introduction. Ensuite nous enlèverons certaines contraintes et nous étudierons le comportement de ce modèle dans sa version la plus optimale (mais différente de l'équation DNF standard) et montrerons que les propriétés de calcul qui nous intéressent sont conservées.

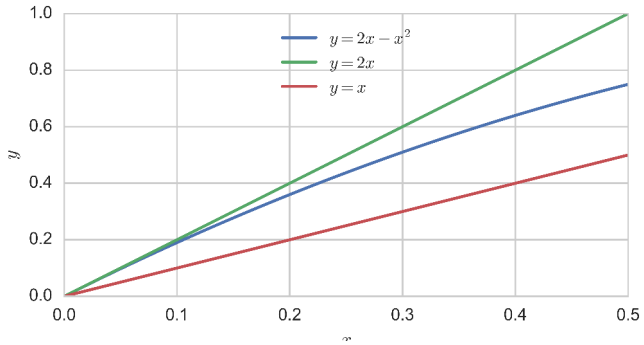


FIGURE 4.2 – Erreurs introduites par l'utilisation de la porte OU pour effectuer une addition $y = x + x$ de flux de bits. Modèle de l'addition avec le OU (bleu), addition avec un multiplexeur (rouge) et addition standard (vert).

4.2 Poids latéraux

Pour conserver un fonctionnement identique au modèle SDNF avec noyau exponentiel, nous allons faire en sorte que l'équation des poids latéraux tende vers une différentielle d'exponentielles.

4.2.1 Formalisation

Pour modéliser aussi précisément que possible le comportement du modèle matériel CASAS, nous allons définir les opérateurs pour transformer un réel $a \in [0, 1]$ en un flux de bits de taille N que nous noterons a_{bs} (bs pour bitstream). Chaque bit de a_{bs} sera noté $a_i, i \in [1, N]$.

La fonction de génération du flux de bits (SNG pour stochastic number generator) est définie comme suit $\forall i \in [1, N], P(a_i = 1) = a$.

La fonction de décodage (SND pour stochastic number decoder) donne une approximation de a, que nous noterons $\hat{a}$, estimée comme suit : $\hat{a} = \frac{1}{N} \sum_{i=1}^{N} a_i$.

Pour différentier l'opérateur d'addition, de l'addition en flux de bits que nous allons utiliser nous utiliserons l'opérateur $\diamondsuit$.

Si un neurone est activé il émet une impulsion sous la forme d'un flux de bits que nous noterons $\delta_{bs} = \delta$ sinon $\delta_{bs} = 0$. Un autre neurone à distance de Manhattan d recevra le flux de bits excitateur

$$E_{bs} = \delta_{bs} A_{bs}^d, \tag{4.1}$$

où A est le poids synaptique de chaque synapse excitatrice. De même, pour la couche inhibitrice avec les synapses de poids B,

$$I_{bs} = \delta_{bs} B_{bs}^d, \tag{4.2}$$

De sorte que les influences latérales pour le neurone à distance d sont en version asymptotique :

$$w(d) = K_e \delta_{bs} A_{bs}^d - K_i \delta_{bs} B_{bs}^d, \tag{4.3}$$

où K_e et K_i sont des constantes du modèle.

Si plusieurs neurones sont activés, il faut prendre en compte l'addition des flux de bits. L'influence latérale reçue par un neurone à la position x sera :

$$w_n = K_e \diamondsuit_{i=1}^{R^2} (\delta_{bs}(i) A_{bs}^{|x-i|}) - K_i \diamondsuit_{i=1}^{R^2} (\delta_{bs}(i) B_{bs}^{|x-i|}), \tag{4.4}$$

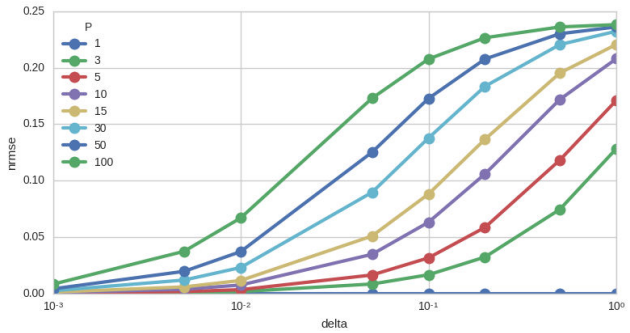


FIGURE 4.3 – Erreurs introduites par l'utilisation de la porte OU pour effectuer une addition de flux de bits. Conséquence sur le modèle d'intégration neuronale des impulsions synaptiques. La NRMSE en fonction de la valeur initiale de l'impulsion δ est indiquée en fonction du nombre de neurones activés P.

où $\Diamond_{i=1}^{n}$ est la somme de flux de bits généralisée définie comme suit (pour la porte OU) en version asymptotique :

$$\Diamond_n a_{bs} = 1 - (1 - a)^n. \tag{4.5}$$

Nous allons pour l'instant supposer que tous les neurones activés sont à la même distance d du neurone et que P neurones sont activés. L'équation devient donc :

$$w_n = K_e \Diamond_P (\delta_{bs} A_{bs}^d) - K_i \Diamond_P (\delta_{bs} B_{bs}^d), \tag{4.6}$$

$$\iff w_n = K_e (1 - (1 - \delta_{bs} A_{bs}^d)^P) - K_i (1 - (1 - \delta_{bs} B_{bs}^d)^P). \tag{4.7}$$

C'est un modèle asymptotique approché qui nous permet d'avoir une idée de la déformation de la courbe des poids latéraux en fonction des différents paramètres. Comme dans le chapitre précédent l'impact des paramètres sur la propagation des influences latérales sera montré avec la racine de l'erreur quadratique normalisée (NRMSE) calculée entre la diffusion obtenue avec une couche de diffusion du modèle CASAS $\bar{D}$ et une diffusion parfaite D.

Sur la figure 4.3 l'erreur quadratique moyenne normalisée est tracée pour différentes valeurs de P et de δ. Il apparaît que plus δ et P sont élevés plus l'erreur est grande. C'est dû à la porte OU qui est précise pour les petites valeurs de flux de bits uniquement. Par conséquent il faut choisir une valeur de départ δ petite pour les impulsions de façon à laisser de la place aux additions dans le flux de bits. La valeur de δ va dépendre du nombre d'activations maximal attendu au même endroit P. Les statistiques effectuées sur les scénarios courants montrent par exemple un maximum de $P = 28$ pour une résolution de $R = 49$. On peut donc se contenter d'une valeur de $\delta = 0.01$. La figure 4.4 montre la forme de la courbe des poids latéraux idéale (DoE) sur la partie gauche (à gauche de l'axe des ordonnées) et la courbe obtenue avec l'addition des flux de bits définie ci-dessus sur la partie droite. On voit que la différence est raisonnable jusqu'à $P = 50$ activations.

4.2.2 Conditions limites

Une modélisation plus réaliste de la diffusion des flux de bits permet d'estimer l'impact des additions lors du routage et d'estimer la précision nécessaire pour les flux. Nous allons pour cela

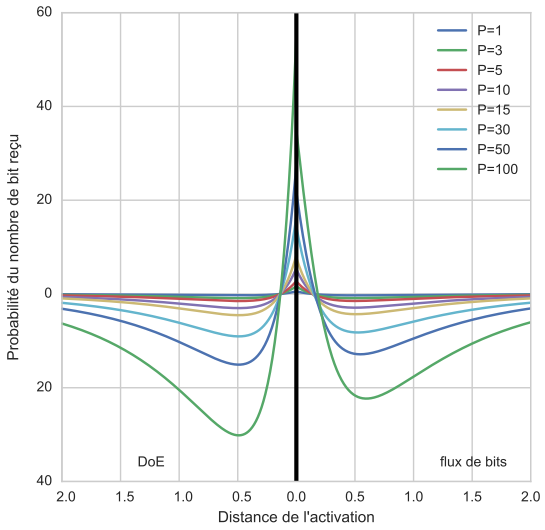


FIGURE 4.4 – Déformation de la courbe des poids latéraux lorsque le nombre d'activations P augmente. À gauche la différence d'exponentielles qui sert de référence ($kE = 1.25, kI = 0.7, pE = 0.0043, pI = 0.3$). À droite une première approximation de la courbe des poids latéraux obtenue avec l'arithmétique stochastique ($\delta = 0.01$).

utiliser un simulateur simplifié basé sur une arithmétique de flux de bits.

Cette simulation logicielle est basée sur une bibliothèque développée en C++ encapsulant les flux de bits dans un vecteur d'entiers non signés. Le but de ce simulateur est de calculer la phase de diffusion des flux de bits en deux étapes de calcul. 1) Les flux de bits sont générés pour les cellules (flux de bits impulsionnels) et les routeurs (flux de bits synaptiques). 2) Les additions et multiplications entre flux de bits sont faites de manière récursive en remontant le graphe de connectivité.

Ce niveau de simulation intermédiaire permet d'étudier l'influence qu'auront les additions de flux de bits au niveau de chaque routeur et au niveau des cellules, par contre il ne permettra pas d'étudier l'influence de la corrélation des nombres aléatoires par exemple.

Sur la figure 4.6 on peut voir la NRMSE qui augmente avec P et δ. Au delà de l'imprécision de l'addition déjà constatée, on voit également l'influence de l'imprécision des flux de bits pour les petites valeurs de δ.

Les nombres pouvant être représentés par un flux de bits de taille n sont tous les rationnels $p = n_1/n$ donc $\delta = 0.01$ peut être représenté par un flux de taille $n = 100$. En revanche le résultat de la multiplication $p\delta$ ne fait plus partie de l'intervalle des rationnels possibles. Il faut donc augmenter la longueur du flux de bits lorsque δ diminue. Plus de détails sur les relations entre les différents paramètres sont présentés dans l'annexe C.

L'affichage d'une diffusion (fig 4.5) montre que non seulement les valeurs sont aplaties par rapport à la DoE voulue, mais qu'il y a aussi une dissymétrie entre une coupe horizontale et une coupe verticale de la matrice des valeurs obtenues. La dissymétrie pour la diffusion dans une carte de R^2 neurones D_{ij} est calculée comme la différence quadratique moyenne entre une coupe

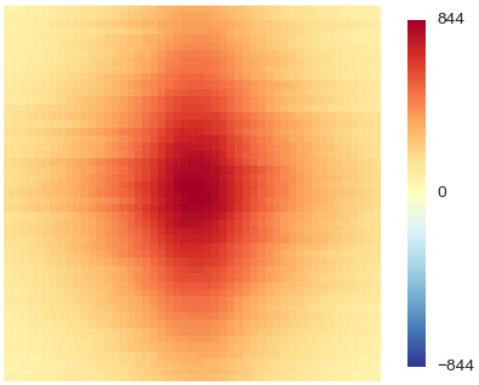


FIGURE 4.5 – Visualisation de l'asymétrie avec $P = 50$, $\delta = 0.05$ et $N = 1000$.

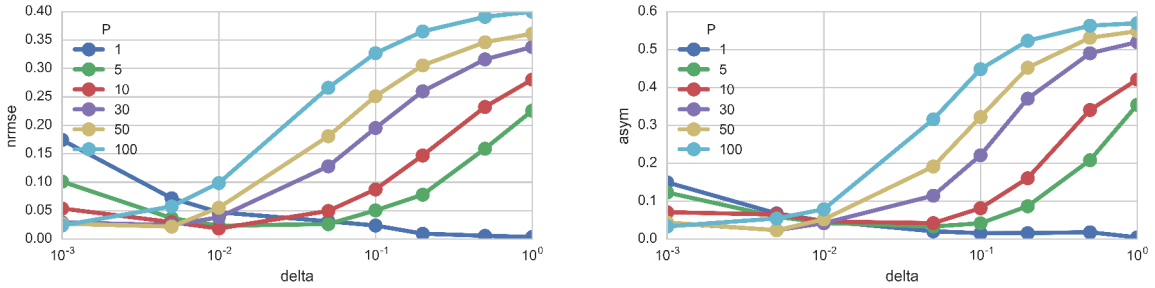


FIGURE 4.6 – Biais de propagation des flux de bits. Haut : la NRMSE en fonction de δ et du nombre d'activations P. Bas : l'asymétrie entre le nombre de bits reçus par les neurones situés dans la colonne du milieu et ceux situés dans la ligne du milieu. $N = 10000$.

horizontale et une coupe verticale passant par le centre.

$$\Gamma = \sqrt{\sum_{i=1}^{R} (D_{i,R/2} - D_{R/2,i})^2} \tag{4.8}$$

On voit sur la figure 4.6 qu'il existe une valeur optimale pour δ afin de minimiser l'erreur avec la DoE et l'asymétrie. On prendra $\delta = 0.01$.

Sur la figure 4.7 on voit aussi qu'il y a une valeur optimale pour la taille du flux de bits N afin que la précision pour $\delta = 0.01$ soit suffisamment bonne.

Nous avons donc réussi à déterminer empiriquement les paramètres pour la fonction des poids latéraux du modèle. La diffusion des informations d'activation par utilisation d'une arithmétique de flux de bits dégradée pour l'addition implique des biais asymptotiques important pour les valeurs de flux élevées. Nous avons montré qu'il est nécessaire de diminuer fortement la valeur de départ des flux de bits $\delta = 0.01$ afin de conserver des résultats asymptotiques approchant la différence d'exponentielles étudiée dans le chapitre précédent.

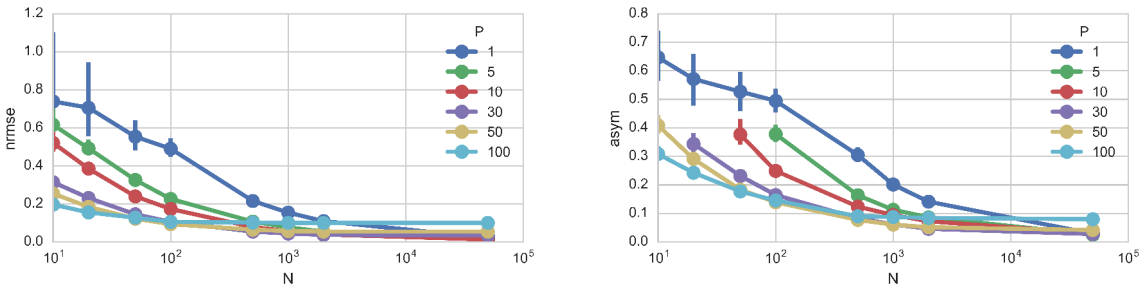


FIGURE 4.7 – Erreur pour différentes tailles de flux N et différents nombres d'activations P. Haut : NRMSE. Bas : asymétrie. $\delta = 0.01$.

Outre ce biais asymptotique le modèle est limité par la précision des valeurs encodées par les flux de bits, précision qui est directement liée à la longueur des flux de bits. Nous avons montré expérimentalement qu'une longueur de $N = 100$ bits est suffisante lorsqu'il y a beaucoup d'activations ($P > 30$), autrement il faut plus de bits (on prendra $N = 1000$).

La simulation de ce modèle permet de s'assurer que le comportement converge lorsque la taille du flux de bit augmente. Ces simulations sont faites sur un réseau de résolution $R = 49$ avec une probabilité impulsionnelle de $\delta = 0.01$ (fig. 4.8).

Ces résultats montrent qu'un flux de bit de taille $N = 1000$ est nécessaire pour le suivi de cible et d'au moins $N = 20000$ pour le scénario de mémoire de travail. Le temps de diffusion est alors calculé comme la taille du flux de bit N plus le temps que le dernier bit traverse la carte $td = N + 2R$.

4.3 Implantation matérielle

L'implantation est très similaire à celle du modèle RSDNF. Nous détaillerons ici les modules qui sont différents.

4.3.1 Architecture

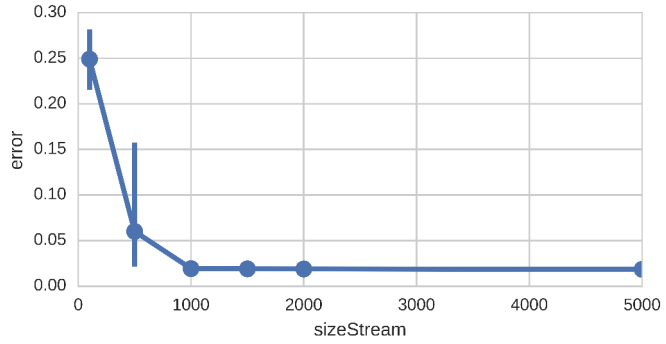
4.3.1.1 Routeur

Les routeurs sont très simplifiés vu qu'il ne sont plus composés que d'une porte OU à 4 entrées et d'une porte ET à deux entrées pour la multiplication avec le poids synaptique (généré par un substrat CAPRNG comme pour RSDNF). Sur la figure 4.9, la source aléatoire est un flux de bits synaptique «SynBs» pour synaptic bitstream en anglais.

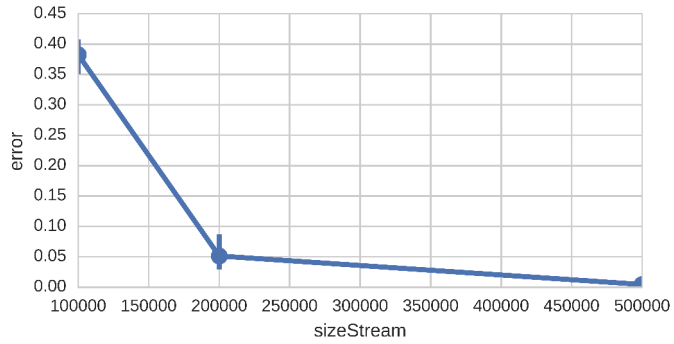
4.3.1.2 Cellule

Une cellule de routage CASAS est composée de la logique de routage (4 routeurs). Le flux de bits impulsionnel (commun à toutes les cellules de routage du neurone) est généré avec un compteur (voir fig 4.10).

Dans l'architecture étudiée, il y a une cellule de routage pour la couche excitatrice et une pour la couche inhibitrice.



(a) Suivi de cible



(b) Mémoire de travail

FIGURE 4.8 – Erreur de distance en fonction de la taille des flux de bits N. Moyenne sur 50 répétitions et intervalles de confiances bootstrap à 95%. En haut, le scénario suivi de cible robuste (ici avec une intensité de bruit de 0.4 et 5 distractions) nécessite beaucoup moins de répétitions pour tendre vers le résultat asymptotique (ligne noire) que le scénario mémoire de travail (bas).

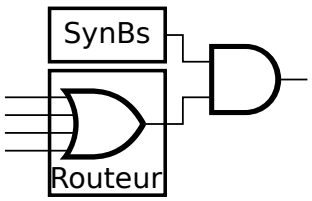


FIGURE 4.9 – Diagramme du routeur CASAS et du flux de bits synaptique associé.

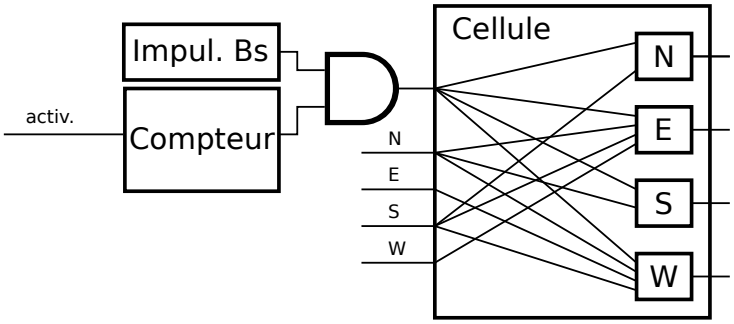


FIGURE 4.10 – Diagramme de la cellule CASAS et du flux de bits impulsionnel associé. Lorsque le neurone est activé, un compteur active le flux de bits impulsionnel pour la longueur des flux de bits désirée. Les impulsions entrantes sont routées avec les quatre routeurs nord (N), est (E), sud (S) et ouest (W).

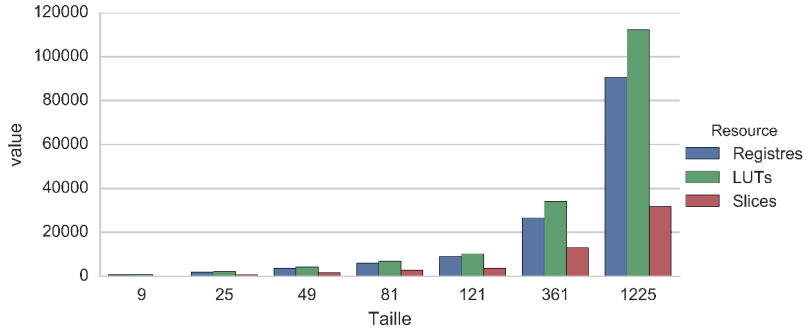


FIGURE 4.11 – Surface d'implantation pour le modèle CASAS.

4.3.2 Résultat d'implantation sur FPGA

On voit sur les figures 4.11 et 4.12 que la surface d'implantation pour le modèle matériel CASAS est beaucoup plus raisonnable que celle de RSDNF.

Le nombre de registres et de LUT est divisé par deux environ. La surface de CASAS se rapproche de la surface de l'implémentation parallèle centralisée «AER» étudiée dans le chapitre 2.

Les architectures RSDNF et CASAS sont comparées ici avec un substrat de génération de nombres aléatoires du même type (CAPRNG) et dédiant 8 bits aléatoires à chaque flux de bits. Le modèle CASAS nécessitant une source aléatoire supplémentaire par neurone (pour générer le flux de bits dédié représentant une impulsion), la surface dédiée au CAPRNG est un peu plus importante que pour RSDNF. Cependant, la technique de réduction de surface en partageant les ressources aléatoires est toujours valide pour CASAS et permet donc de diviser la surface dédiée à la génération des nombres aléatoires par 9 pour CASAS et 8 pour RSDNF. Cela nous ramène à une surface équivalente dédiée à la génération des nombres aléatoires pour les deux modèles. La figure 4.13 compare la surface d'implémentation de CASAS et RSDNF avec les différentes techniques d'optimisation de la génération de nombres aléatoires.

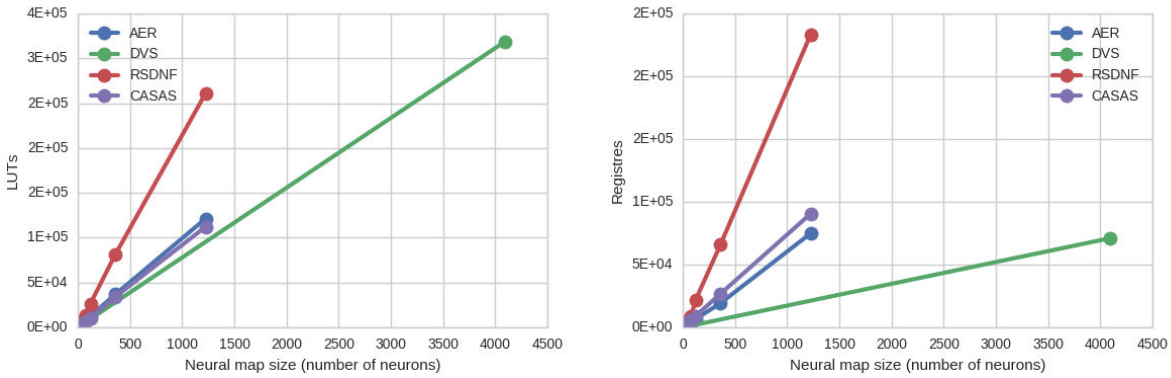


FIGURE 4.12 – Résultats d'implantation du modèle CASAS sans optimisation comparé au modèle RSNDF et aux deux architectures centralisées basées sur le protocole AER.

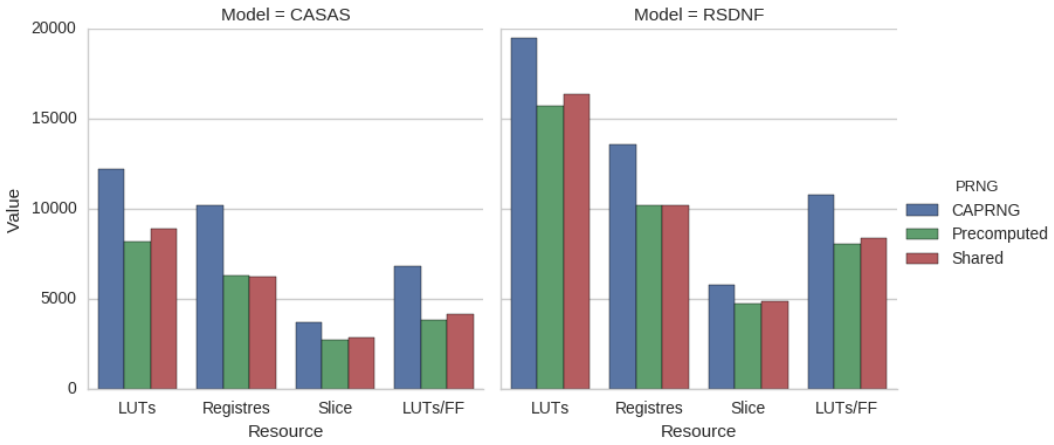


FIGURE 4.13 – Surface d'implémentation de 81 neurones avec les modèles matériels RSDNF et CASAS et en utilisant les variantes liées à la production des nombres aléatoires.

TABLE 4.1 – Registres associés à chaque module avec leur taille.

Module	Registre	Taille	Description
CellBsRsdnf	spike bs	1	Flux de bits généré lors d'une activation
CellBsRsdnf	nb bits received	10	Nombre de bits reçu par la cellule
CellBsRsdnf	activated	1	Haut si le neurone est activé
CellBsRsdnf	nb bits to gen	11	Compteur de la taille du flux de bits
Router	bs out	1	Flux de bits en sortie du routeur

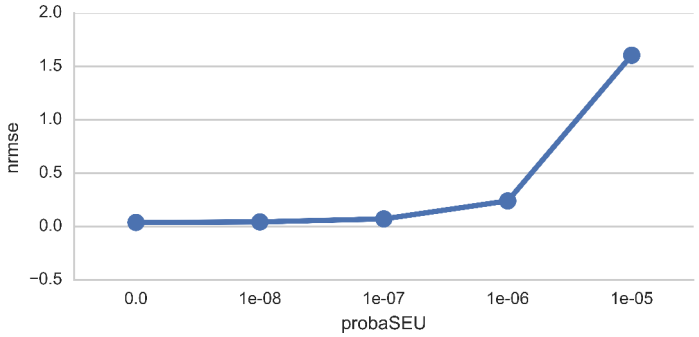


FIGURE 4.14 – NRMSE pour différents taux d'inversion de bit.

4.4 Robustesse matérielle

Le modèle matériel CASAS contient moins de registres que RSDNF (64827 registres contre 132055 pour RSDNF). Statistiquement les erreurs affecteront donc moins les registres (voir table 4.1). Nous allons quand même étudier l'impact des fautes temporaires ou permanentes sur ces registres.

Le nombre de registres à plusieurs bits encodant pour un nombre en base binaire est aussi moins important que pour l'architecture RSDNF. Il y a les registres d'accumulation des bits reçus et les compteurs qui génèrent les flux de bits lors qu'il y a activation.

Pour les simulations la résolution est de $R = 49$, la probabilité $p = 0.93$, il y a 10 neurones activés au milieu de la carte, $\delta = 0.01$ et la longueur des flux de bits $N = 2000$. Le temps de propagation est donc de $N + 2R = 2098$.

4.4.1 Erreurs temporaires

Sur la figure 4.14 on voit que les conséquences de plusieurs inversions de bits lors d'une phase de propagation est un peu plus limité que dans le cas du modèle matériel RSDNF. Ici l'erreur de diffusion est importante à partir de $p = 1 \times 10^{-6}$ contre $p = 5 \times 10^{-7}$ pour RSDNF.

4.4.2 Erreurs permanentes

La figure 4.15 montre l'erreur lors d'une phase de routage avec une erreur permanente de type «1» ou «0».

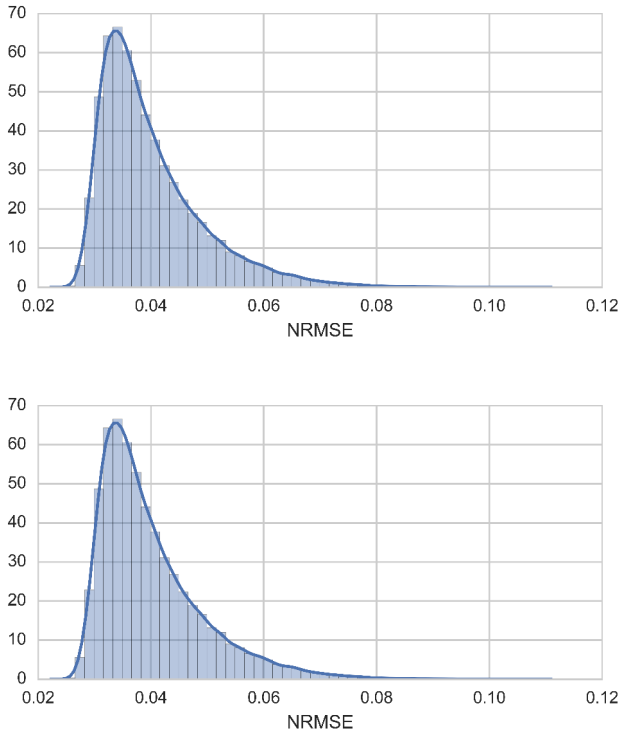
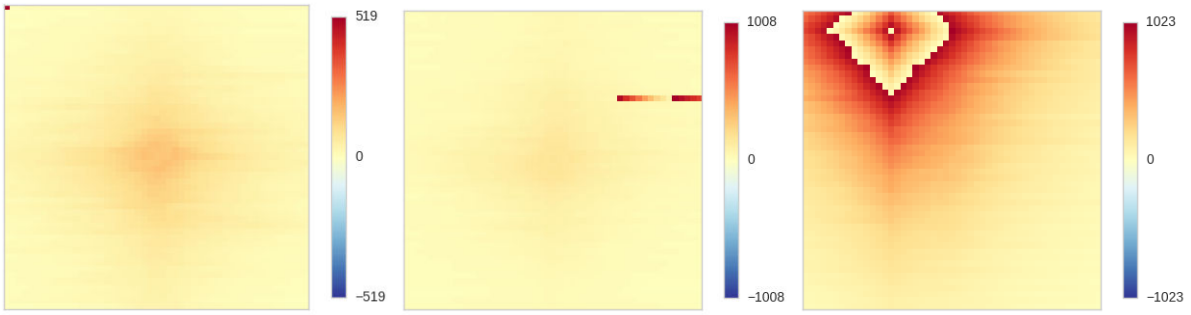


FIGURE 4.15 – Histogramme des erreurs permanentes sur une phase de propagation dans CASAS. En haut, les bits sont forcés à «1» les uns après les autres et en bas ils sont forcés à «0».



(a) $e = 0.07$ (b) $e = 0.3$ (c) $e = 2$

FIGURE 4.16 – Exemple d'erreurs permanentes sur une phase de propagation. Les deux premières erreurs sont légères puisque plutôt localisées. La dernière est plus grave.

Pour les erreurs de type «1», les résultats sont très différents de ceux du modèle RSDNF. Ici 18% des fautes seulement ont une erreur $e > 0.1$ (contre 44% pour RSDNF). Par contre les erreurs ont potentiellement un impact beaucoup plus fort (l'erreur maximale était $e = 0.5$ pour RSNDF et est de 2.5 pour CASAS). Ceci est dû à la longueur plus importante de la phase de propagation $t_d = 2098$ contre 298 pour RSDNF. Par conséquent les fautes générant un flux de bits toujours actif ont plus de temps pour perturber les comportements. Sur la figure 4.16 quelques exemples sont montrés. On voit ici l'impact de la taille des registres sur les fautes : les registres du nombre de bits reçus qui sont montrés dans ces images sont limités à 10 bits et leur valeur est donc limitée à 1023. Les vagues de diffusions correspondent à la remise à «0» de ces registres lorsque la valeur limite du registre est dépassée. Réduire la taille des registres pourrait non seulement diminuer le nombre d'erreurs affectant les registres mais pourrait aussi limiter les conséquences de ces erreurs.

Cette section nous a donc permis de montrer que le modèle matériel CASAS est plus résistant aux fautes que le modèle RSDNF sur les registres. La première raison est que le nombre de registres sensibles est moins important. Les registres comptant le nombre de bits reçus ne sont pas critiques puisqu'ils ont un impact sur un seul neurone. Les registres les plus critiques sont ceux dont dépend la génération des flux de bits (le compteur, le signal d'activation et les flux de bits en sortie des routeurs).

Les conséquences sur le comportement du DNF sont similaires au chapitre précédent. Si la valeur du potentiel synaptique de plus d'un neurone est trop forte, cela perturbe irrémédiablement le comportement du DNF.

Même si le nombre de fautes graves est moindre, il y a encore un nombre trop important de fautes qui peuvent perturber complètement le comportement du DNF.

Avant de terminer sur une comparaison multi-critères des différents modèles et implémentations proposés, nous allons étudier rapidement une variante possible de CASAS

4.5 Version fréquentielle du modèle CASAS

Nous avons étudié dans ce chapitre un modèle d'architecture matérielle basé sur la version impulsionnelle des DNF.

Les résultats obtenus sur l'influence des différentes variables sur l'erreur lors de la propagation des informations latérales ont montré que le nombre de neurones activés P améliore la qualité

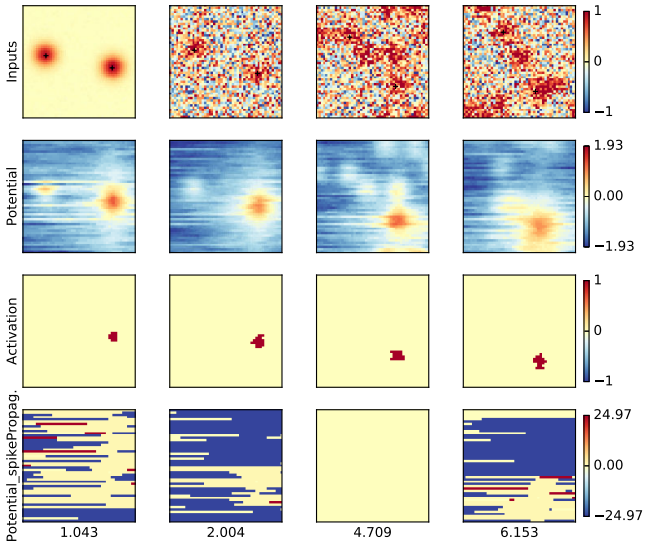


FIGURE 4.17 – Six secondes de simulation de la variante fréquentielle du modèle CASAS.

de l'estimation car le nombre de flux de bits dont la valeur est non nulle est plus important. L'estimation reçue par chaque neurone des activités latérales est donc plus proche de la valeur asymptotique.

De plus le temps de propagation dépend uniquement de la longueur des flux de bits N. La longueur du flux de bits nécessite l'utilisation de compteurs servant à stopper le flux de bits une fois que N itérations sont écoulées.

On peut donc se demander si la version fréquentielle des champs neuronaux dynamiques (la version initiale proposée par Amari) avec une fonction d'activation de type tout ou rien (heaviside) n'est pas plus adaptée à la transmission par flux de bits du modèle CASAS.

Nous allons étudier dans cette section une telle variante basée sur les DNF fréquentiels.

4.5.1 Principe général

Un DNF de ce type a une activation binaire mais continue dans le temps. Tant qu'un neurone est actif il génère un flux de bits de valeur δ. On voit par exemple sur la figure 4.17 que l'activation des neurones n'entraîne pas de remise à zéro du potentiel membranaire des neurones et que l'activation des neurones dure alors pendant plusieurs itérations.

L'avantage principal de ce modèle est d'enlever le compteur et ainsi d'économiser de la surface et d'améliorer la résistance aux fautes matérielles.

Il faut cependant régler les paramètres de façon à obtenir le comportement désiré. Les paramètres dépendent de la fréquence de mise à jour des neurones et du générateur du flux de bits. Pour simplifier on suppose que l'horloge de la mise à jour du potentiel membranaire et du générateur de flux de bits est la même de façon à éviter le stockage du nombres de bits excitateurs et inhibiteurs reçus au sein de chaque cellule.

Plusieurs problèmes sont posés par cette architecture. Comment choisir δ ? Comment calculer la fuite du potentiel membranaire ? En effet, si la fréquence de mis à jour du neurone est très

rapide (toutes les μs par exemple) alors la fuite sera de l'ordre du millionième du potentiel synaptique courant. Les besoins en précisions sont donc différents.

4.5.2 Choisir δ

Comme décrit au cours de ce chapitre, δ doit être choisi de sorte que l'asymétrie soit minimale (différence entre la diffusion horizontale et verticale) et de sorte à ce que le bruit dû à la diffusion aléatoire soit aussi minimal.

Pour choisir δ nous allons reprendre les résultats des sections précédentes. On peut exprimer les résultats de la section précédente de manière légèrement différente. Les paramètres que nous avons choisis ont deux contraintes : l'asymétrie due à la mauvaise qualité de l'addition, la précision dépendant du nombre d'activations A et de la longueur du flux de bits N.

Ici N sera significativement plus grand que dans les sections précédentes. En effet, les flux de bits seront générés à très grande vitesse (par exemple avec une période de $1\mu s$). En utilisant le même scénario de suivi de cible avec la même constante temporelle τ, les neurones restent actifs pendant plusieurs dixièmes de millisecondes. Si l'on prend une fenêtre temporelle de 0.1s la longueur du flux de bits sera donc de $N = 1 \times 10^5$.

Par conséquent, la latitude est plus grande pour choisir δ puisque plus le flux de bits est long plus il est précis. On choisira donc δ entre 1×10^{-3} et 1×10^{-4}.

4.5.3 Précision des neurones

La faible période de mise à jour impose une précision plus grande pour les neurones.

4.5.3.1 Fuite membranaire

Par exemple si on considère la précision nécessaire pour calculer la fuite avec $dt = 1 \times 10^{-6}$ et $\tau = 0.5$, il faut diviser le potentiel membranaire par $\frac{\tau}{dt} = 5 \times 10^5$. Il faut alors modifier τ ou dt de façon à pouvoir faire la division avec un décalage de 19 bits. La précision nécessaire pour encoder le potentiel sera donc supérieure à 20 bits.

4.5.3.2 Flux de bits latéraux

Le potentiel latéral excitateur reçu par un neurone dans une fenêtre temporelle est calculé à partir des paramètres de la différence d'exponentielles (DoE).

En prenant les paramètres de la DoE utilisée pour le suivi robuste de cible pour la couche inhibitrice $k_i = 0.6$. Cette valeur normalisée est transformée en valeur réelle avec $k_i' = 160\frac{dt}{\tau}\frac{k_i}{R^2 * \delta N}$ avec $R = 49$, $\delta = 0.01$, $n = 1$ et $dt = 1 \times 10^{-6}$ et $\tau = 0.5$, $k_i' \approx 8 \times 10^{-6}$.

Donc il faut minimum 17 bits dans la partie fractionnelle pour encoder cette valeur.

Il est évidement possible de diminuer la valeur de δ pour augmenter cette valeur minimale.

Cette version échange donc les compteurs et les registres stockant les influences latérales par une précision plus grande pour le potentiel membranaire. L'économie est importante puisque 21 bits de stockages sont gagnés.

La résistance aux fautes des registres n'a pas été étudiée pour cette version mais il est évident que si un bit est toujours actif à la sortie d'un routeur les conséquences seront aussi graves que pour CASAS.

Par contre le modèle sera beaucoup plus stable grâce à la longueur des flux de bits bien supérieure à celle du modèle CASAS.

TABLE 4.2 – Bilan comparatif des différent modèles matériel introduits.

	AER	RSDNF	CASAS
Stabilité du comportement	+	-	–
Surface	+	-	+
Vitesse	++	-	- -
Robustesse	-	- -	-
Décentralisation	-	+	+
Asynchronie	- -	-	+

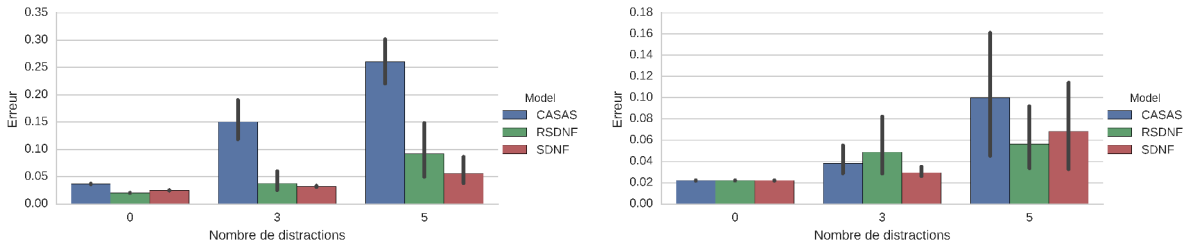


FIGURE 4.18 – Robustesse du modèle CASAS comparé aux modèles RSDNF et SDNF. Avec les paramètres optimisés pour RSDNF à gauche et avec les paramètres optimisés pour CASAS à droite.

4.6 Bilan comparatif

La table 4.2 compare les différentes caractéristiques des modèles matériels présentés dans ce manuscrit. Grossièrement, les modèles CASAS et RSDNF sacrifient la surface ou la vitesse au profit d'un calcul décentralisé. Le modèle CASAS sacrifie encore plus de vitesse au profit d'un calcul asynchrone.

L'étude limitée aux registres de la robustesse aux fautes matérielles n'est pas suffisante pour conclure de façon absolue. Nous avons cependant montré que le modèle CASAS était plus robuste aux fautes matérielles touchant les registres et que ces modèles n'ont pas une robustesse suffisante pour des réductions de coût de production.

4.6.1 Stabilité du comportement

Le modèle matériel AER est évidement beaucoup plus stable que les modèles cellulaires vu qu'il n'y a pas de composante aléatoire.

Comme on l'a vu, la stabilité des modèles RSDNF et CASAS dépend de la longueur de leur flux de bits. En choisissant $\delta = 0.01$, les résultats expérimentaux montrent qu'une longueur de flux de bits 100 plus grande pour CASAS que pour RSDNF suffit à obtenir le même comportement pour les scénarios de suivi ou de mémoire.

Cependant, il apparaît que le scénario de suivi robuste est moins performant pour CASAS (figure 4.18).

Ceci est dû au fait que le calcul des interactions latérales est biaisé lorsqu'il y a un grand nombre d'activations. Le modèle CASAS est donc plus sensible aux activations atypiques, par

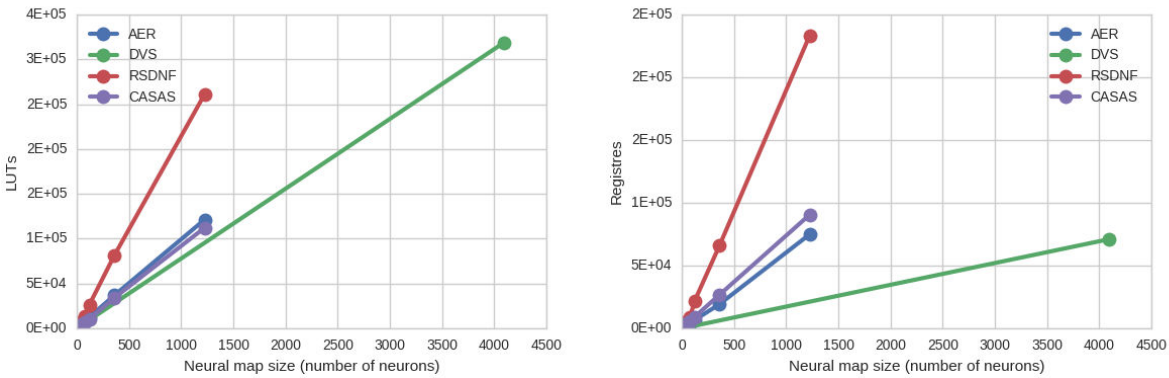


FIGURE 4.19 – Résultats d'implantation du modèle CASAS comparé au modèle RSNDF et aux deux architectures centralisées basées sur le protocole AER.

exemple dans le cas où des distractions sont très proches de la bulle d'activité principale.

Les résultats dépendent beaucoup des paramètres utilisés. Il est donc souhaitable d'optimiser les paramètres avec le modèle matériel plutôt que le modèle approché. Dans ce cas il est possible de trouver des paramètres satisfaisants pour CASAS qui satisfassent aussi d'autres modèles (voir figure 4.18). Ces paramètres sont donc plus robustes à la déformation des interactions latérales engendrée par l'addition des flux de bits et permettent l'utilisation de flux de bits plus courts.

Il est donc conseillé de ré-optimiser les paramètres en utilisant le modèle matériel afin de profiter de performances maximales de celui-ci. L'inconvénient est que la simulation du modèle matériel est plus longue que celle du modèle approché, l'optimisation requiert donc plus de ressources.

4.6.2 Surface d'implémentation

La surface d'implémentation pour CASAS est beaucoup moins importante que pour RSDNF et a peu près équivalente à AER (voir fig. 4.19).

4.6.3 Vitesse de mise à jour

Pour que la vitesse soit comparable il faut étudier le temps mis par l'implémentation pour simuler 1s. Il faut donc prendre en compte à la fois la fréquence maximale de l'implémentation et le temps de diffusion entre chaque mise à jour du potentiel. On peut voir sur la figure 4.20, la fréquence maximale pour les trois modèles avec différentes résolutions. On prend $P = 10$ (nombre d'activations) et $R = 9$ (résolution). Pour RSDNF le temps de diffusion est de $t_d = NP + 2R = 218$ (avec $N = 10$). Il faut donc 218 itérations pour assurer la diffusion des informations entre chaque mise à jour des potentiels. Pour CASAS le temps de diffusion est de $N + 2R = 2018$ en prenant $N = 2000$.

La fréquence maximale est de 154 MHz pour RSDNF et 235 MHz pour CASAS, la fréquence de mise à jour du potentiel est donc de 0.71 MHz pour RSDNF et de 0.12 MHz pour CASAS. Le même calcul pour $R = 35$ et $P = 15$ donne une fréquence de mise à jour du potentiel de 0.26MHz pour RSDNF et 0.06 pour CASAS. Le modèle RSDNF est donc 6 fois plus rapide pour $R = 9$ et 4 fois plus rapide pour $R = 35$.

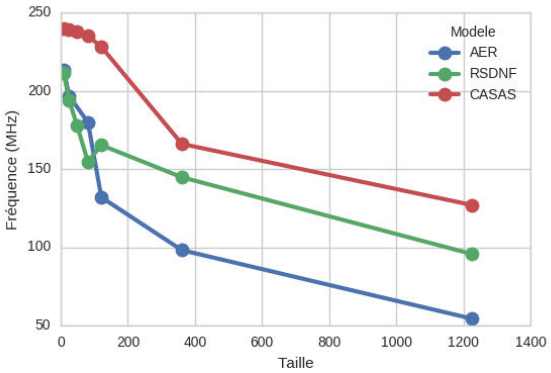


FIGURE 4.20 – Fréquence maximale de l'horloge sur FPGA pour les différents modèles en fonction du nombre de neurones.

La différence de vitesse s'atténue donc avec la résolution grâce à la surface de CASAS qui est beaucoup plus faible et autorise donc des vitesses d'horloge supérieure.

La fréquence maximale AER est plus faible que CASAS et RSDNF à cause du bus partagé. Mais le temps de propagation est de P pour AER contre $N + 2R$ pour RSDNF. Ce qui donne une fréquence de mise à jour du potentiel de 17.0MHz pour $R = 9$ et 3.63 pour $R = 35$.

Le modele AER est donc 24 fois plus rapide que RSDNF pour $R = 9$ et 13 fois plus rapide pour $R = 35$. Ce rapport décroissant est cohérent avec le fait que les modèles RSDNF et CASAS ont été conçus selon une approche décentralisée et cellulaire qui induit une mise à l'échelle aisée. L'avantage en vitesse des implantations à base d'AER pourrait donc largement s'amenuiser pour des tailles élevées de DNF.

4.6.4 Robustesse aux fautes matérielles

Pour la robustesse aux fautes, le modèle AER possède des bus de transfert des événements et un arbitre qui sont des point de fragilité de l'architecture. Nous n'avons pas pu étudier les fautes dans les LUTs ou le routage donc cette fragilité n'a pas pu être évaluée. En revanche les fautes des registres du modèle AER ne peuvent affecter qu'un seul neurone car il n'y a pas de propagation. Par conséquent le modèle AER est beaucoup plus robuste aux fautes affectant les registres que les modèles cellulaires que nous avons étudié. Le seul registre sensible est celui de l'arbitre qui est de la taille du nombre de neurones. Ce registre sauvegarde l'activation des neurones pour future publication. Une faute permanente dans ce registre peut perturber la période de publication en publiant toujours le même neurone.

Le modèle CASAS est plus robuste que RSDNF grâce à la disparition des tampons de stockage des impulsions latérales.

En ce qui concerne la sensibilité aux fautes dans les LUTs, même si celles-ci n'ont pas été étudiées, on peut noter que dans le cas AER, lorsque ces fautes affectent les parties centralisées (bus et arbitre), qui représentent 30% des LUTs de l'implantation, elles peuvent induire le non fonctionnement complet du modèle, ce qui n'est pas le cas pour les versions décentralisées pour lesquelles elles se traduisent par des performances altérées.

4.6.5 Décentralisation

Le caractère décentralisé d'une architecture exprime le fait qu'il n'y a pas de contrôleur central qui donne des ordres aux unités subalternes. Les architectures cellulaires que nous avons présentées ne sont contrôlées que par des horloges. Au contraire l'implémentation AER utilise un arbitre central qui décide des événements AER à publier. Cette architecture est donc centralisée.

4.6.6 Asynchronisme

Les systèmes cellulaires sont traditionnellement définis de manière synchrone. C'est à dire que l'état de toutes les cellules est mis à jour au même moment. Ceci a l'avantage de simplifier les interactions entre les éléments. Ce modèle de mise à jour est aussi compatible avec les architectures matérielles séquentielles qui sont souvent définies de manière synchrone avec une horloge globale synchronisant le changement d'état des bascules. En revanche dans notre recherche sur des substrats de calculs non conventionnels, il est souhaitable de se débarrasser de cette horloge globale pour plusieurs raisons : a) Les contraintes de synchronisation sur ce signal parcourant l'ensemble de la carte matérielle ralentissent fortement la fréquence de cette horloge et donc la vitesse globale de l'implémentation. b) L'horloge constitue un point de vulnérabilité. Si l'horloge ou les routes qu'elle emprunte sont défaillantes alors une grande partie de l'architecture sera dysfonctionnelle.

L'asynchronisme est une propriété beaucoup étudiée dans les automates cellulaires par exemple [Fates, 2013] et elle a été étudiée dans le cadre des champs neuronaux dynamiques [Taouali et al., 2010]. Ces résultats montrent qu'un DNF asynchrone se rapproche beaucoup d'un DNF synchrone avec du bruit. Il est donc tout à fait possible d'envisager un modèle matériel asynchrone basé sur les DNF.

Nous appellerons un modèle asynchrone tout modèle qui ne nécessite pas de synchronisation globale.

4.6.6.1 Synchronisation dans le modèle RSDNF

Dans le modèle RSDNF il est nécessaire d'avoir un signal de synchronisation à cause de l'asymétrie de la vitesse de diffusion.

L'asymétrie de vitesse de diffusion des qImpulsions est provoquée par la charge des tampons qui est plus importante pour les routeurs est et ouest que pour les routeurs nord et sud. Le nombre de qImpulsions arrivant à une distance donnée étant plus important pour la couche inhibitrice que pour la couche excitatrice, le retard est plus important pour la couche inhibitrice.

C'est pour cette raison qu'il est nécessaire d'attendre que les deux couches aient fini la transmission des impulsions avant de mettre à jour le potentiel. Pour cela nous avons utilisé un signal de synchronisation global donc la fréquence est basée sur une estimation du temps de diffusion des qImpulsions inhibiteurs dans le pire cas.

D'autres solutions sont possibles. Il est possible d'utiliser un protocole de synchronisation décentralisée. Cela rendrait le modèle plus complexe mais permettrait une asynchronie des éléments de calcul. Il est donc envisageable d'avoir une implémentation matérielle où chaque cellule a sa propre horloge. Mais le temps de propagation plus le temps de synchronisation rendraient ce modèle peu compétitif en termes de vitesse.

De plus, ce besoin de synchronisation a le défaut majeur de multiplier le nombre de registres qui prennent de la surface et qui sont sensibles aux fautes (voir chapitre précédent).

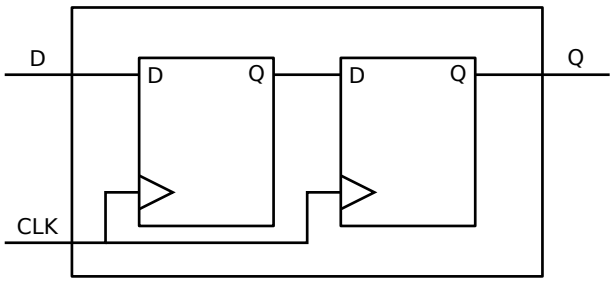


FIGURE 4.21 – Synchronisation du signal «D» en entrée avec l'horloge locale «CLK».

4.6.6.2 Asynchronisme dans le modèle CASAS

Le principal avantage du modèle CASAS par rapport à RSDNF est l'économie des tampons stockant les qImpulsions en cours de transfert. Cette économie se traduit par une économie de surface mais elle a d'autres avantages. Le principal avantage est l'homogénéité de la vitesse de diffusion. C'est à dire que les informations (flux de bits ou qImpulsions) partant d'un neurone quelconque arrivent en même temps au niveau des neurones situés à distance d. On a vu que ces informations étaient dégradées de façon différente lors des transferts horizontaux ou verticaux à cause de l'addition de flux de bits. La propagation n'est donc pas homogène en termes de valeurs propagées mais homogène en termes de vitesse.

L'homogénéité de la vitesse de diffusion est suffisante pour pouvoir implémenter ce modèle de façon asynchrone.

Le potentiel des neurones peut donc être mis à jour à n'importe quel moment sans que cela change le potentiel final du neurone si la discrétisation temporelle est suffisamment fine. Par conséquent il est tout à fait envisageable de mettre à jour le potentiel dès qu'un bit est reçu. En revanche, si on considère une version asynchrone de CASAS à discrétisation temporelle équivalente, il est probable que l'intégration des spikes au fur et à mesure de leur réception va induire des changements comportementaux difficiles à anticiper, comme des saturations potentielles des zones excitées ou au contraire une convergence plus rapide du modèle dans différents scénarios. Les perspectives d'une telle approche asynchrone sont larges, et méritent une étude très approfondie.

Le modèle CASAS est donc compatible avec les architectures matérielles asynchrones ce qui permet une robustesse et une vitesse accrue grâce à la disparition de l'horloge globale. Le passage à une version asynchrone nécessite cependant plusieurs modifications. Si on suppose que chaque cellule a une horloge locale de fréquence identique mais dont la phase est libre, il est nécessaire de synchroniser les entrées de la cellule avec l'horloge locale de la cellule.

Pour cela il est possible d'utiliser un module de synchronisation avec un registre à décalage à deux étapes. La deuxième bascule est ajoutée en cas de métastabilité de la première (voir figure 4.21). Avec 8 connexions latérales par neurone, le surcoût de la version asynchrone est donc de 16 bascules par neurone.

4.6.7 Discrétisation temporelle

Tous au long de ce manuscrit nous avons appelé dt le pas de discrétisation temporelle de l'équation des champs neuronaux dynamiques. Sauf indication contraire nous avons toujours

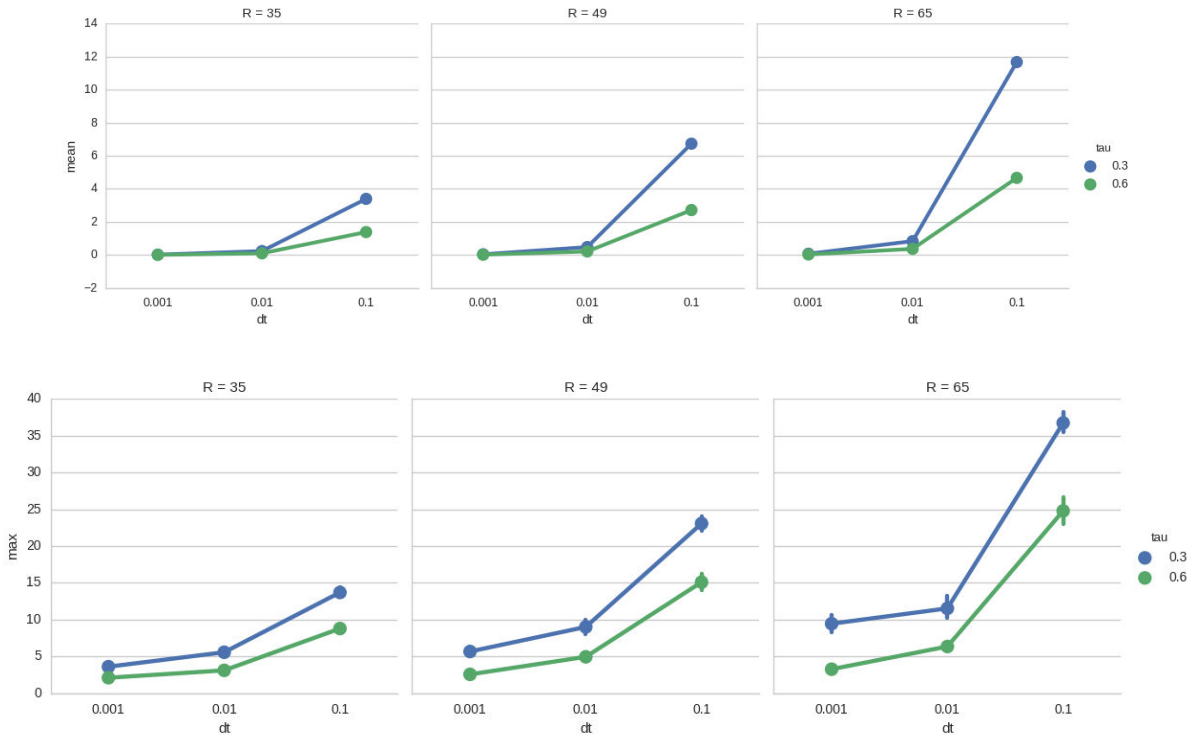


FIGURE 4.22 – Moyenne du nombre d'activations (haut) et nombre d'activations maximal (bas) lors d'un scénario de suivi de cible robuste (bruit=0.4, 3 distractions) avec une résolution temporelle dt, une résolution spatiale R et une constante temporelle τ croissants.

utilisé $dt = 0.1$s pour les simulations.

Nous avons de plus montré comment il était parfois nécessaire de modifier un peu la valeur de dt et τ afin de pouvoir faire la division nécessaire à la fuite membranaire des neurones à l'aide d'un décalage de bits.

Il est donc utile d'étudier l'influence de dt sur les différents aspect de nos architectures. Diminuer dt dans un modèle impulsionnel va nécessairement diminuer le nombre d'impulsions simultanées P. On peut donc se demander si diminuer dt afin de diminuer P n'avantage pas la vitesse des architectures présentées.

4.6.7.1 Influence de la discrétisation sur le nombre d'activations

La figure 4.22 montre l'influence de la résolution R de dt et de τ sur le nombre d'activations moyen et maximal par pas de temps.

Tous les paramètres ont une influence linéaire sur le nombre d'impulsions. Lorsque dt est suffisamment petit ($dt = 1 \times 10^{-3}$) on note que le nombre d'activations simultanées est en moyenne bien inférieur à 1 avec un maximum de $P = 1$ quand $R = 35$, et $\tau = 0.6$. Il est donc possible d'assurer que P ne dépasse quasiment jamais 1.

4.6.7.2 Compromis sur dt

Est-il avantageux de diminuer dt et si oui à quel point ?

Nous avons vu dans ce chapitre que la valeur de δ dépend principalement du nombre d'activations. En effet, un nombre élevé d'activations nécessite un δ petit afin de garder des valeurs de flux de bits faible. Par conséquent la taille du flux de bits doit être élevée ce qui ralentit l'architecture. Si le nombre d'activations P est faible, alors δ sera plus élevé et la taille du flux de bits plus faible.

Par exemple avec $R = 49$ et $P = 5$ il est possible de choisir $\delta = 0.1$ (voir figure 4.6) et donc une taille de flux de bit N entre 100 et 200. La vitesse est donc multipliée par 10.

Mais pour assurer que $P = 5$ il faut diminuer dt par 10 (à condition que τ soit élevé, sinon il faut diminuer encore plus). L'intérêt est donc limité. Le seul cas où l'on peut préférer diminuer dt c'est si l'on souhaite obtenir un modèle asynchrone. Dans ce cas on pourra alors augmenter δ et diminuer N en fonction des résultats.

Pour RSDNF le temps de diffusion dépend plus directement de P. On calcule le temps de diffusion avec $td = NP + 2R$ qui assure que toutes les qImpulsions sont routées. Ici le nombre d'activation ne diminue pas assez vite quand dt diminue pour qu'il soit rentable de diminuer dt.

Pour le modèle AER le temps de diffusion dépend uniquement de P, les mêmes conclusions sont donc valides.

Un cas extrême intéressant conceptuellement est le cas ou dt est suffisamment faible pour être quasiment sûr qu'il y a une seule activation à la fois. On retrouve alors le postula initial du protocole AER (les chances pour que deux événements arrivent en même temps doivent être faible) et cela permettrait de ne plus avoir de temps de diffusion pour le modèle AER.

Avec ce postula le modèle RSDNF n'aurait pas besoin de tampon, le modèle CASAS pourrait avoir $\delta = 1$ et ces deux modèles deviendraient donc équivalents.

4.6.8 Compromis pour l'addition

Il est possible d'augmenter la précision des additionneurs de CASAS en ajoutant un registre à la porte OU. Cette idée est utilisée par exemple dans [Patel et al., 2007b], qui utilise cette propriété afin d'améliorer la qualité du réseau de neurones implanté.

L'idée consiste à stocker dans le registre la retenue de l'addition de deux bits hauts afin de la réinjecter dans le flux plus tard.

Le modèle RSDNF est en fait la version extrême de cette optimisation puisqu'on s'attache à avoir une addition parfaite des flux de bits quitte à les rallonger avec le stockage des bits dans un tampon.

On peut donc chercher un compromis entre CASAS et RSDNF, cependant les tampons provoquent un délai de propagation horizontale qui n'est pas souhaitable.

* * *

Dans ce chapitre, nous avons présenté le modèle CASAS-DNF. L'arithmétique à flux de bits permet de beaucoup simplifier l'architecture puisque les multiplications et les additions sont remplacées par des portes ET et OU respectivement. L'utilisation des portes OU pour les additions diminue cependant artificiellement la valeur des potentiels latéraux et ce de façon asymétrique. Nous avons montré comment conserver des petites valeurs tout au long du routage des informations latérales afin d'obtenir un comportement proche du modèle RSDNF.

Ceci nécessite cependant l'utilisation de flux de bits de longueur assez élevée en pratique. Le temps dédié à la diffusion est donc supérieur à celui du modèle RSDNF mais a l'avantage d'être défini de façon précise.

Nous avons aussi montré que ce modèle était tout à fait adapté à une architecture asynchrone et nous avons discuté des avantages de l'utilisation ou non de la version impulsionnelle des champs

neuronaux dynamiques. Un modèle asynchrone à champs neuronaux non impulsionnels semble également prometteur en termes de surface, de vitesse et de robustesse.

Dans tous les cas la robustesse de ces modèles matériels n'est pas parfaite puisqu'une activation permanente d'un flux de bits suffit à propager des valeurs aberrantes en cascade dans toute la carte. L'identification de cette faiblesse peut motiver l'étude d'une variante qui serait moins sensible à de telles valeurs aberrantes, par exemple au travers de l'introduction de mécanismes de saturation.

Le comportement aléatoire du modèle CASAS-DNF, tout comme celui de RSDNF, rend l'utilisation de scénarios à mémoire de travail difficile à cause de la sensibilité de ce type de scénario aux informations latérales. En effet la mémoire de travail est un état d'équilibre basé sur les interactions latérales. Des valeurs trop éloignées de la valeur asymptotique peuvent facilement déstabiliser cet état. Il faut donc un temps de propagation très long pour atténuer le bruit des interactions latérales que nous introduisons dans ces modèles.

Conclusion

Dans la recherche de nouveaux paradigmes de calcul orientés vers des implantations matérielles permettant de développer des architectures intelligentes nous avons présenté plusieurs propositions pour l'implémentation cellulaire des champs neuronaux dynamiques. Nous avons vu que les champs neuronaux dynamiques (DNF) sont un bon modèle d'approximation du comportement des colonnes corticales et dont les propriétés mathématiques sont intéressantes pour des domaines liés au calcul embarqué : vision, contrôle, planification, etc.

Nous avons présenté quelques implémentations possibles des DNF sur FPGA, prenant avantage des paradigmes de calcul impulsionnels orientés événements, plus proche du calcul neuronal, et surtout plus économiques en termes d'énergie. C'est dans ce cadre que nous avons proposé une architecture événementielle compatible avec les systèmes neuromimétiques et permettant de réaliser le calcul émergent des DNF avec une carte FPGA de taille raisonnable et en profitant du multiplexage temporel.

Mais une telle architecture, même si elle permet d'utiliser quelques cartes FPGA pour faire des architectures d'assez haut niveau pour des applications embarquées, ne change en rien le paradigme de calcul introduit par von Neuman. C'est d'ailleurs le cas de beaucoup d'architectures neuromimétiques. Les architectures les plus performantes et les plus importantes s'appuient toujours sur un calcul numérique basé sur la séparation de la mémoire et des calculs. Certes le nombre de cœurs de calcul est énorme (SpiNNaker par exemple avec 1×10^6 processeurs ARM) mais le routage devient de plus en plus difficile jusqu'à éventuellement stopper la mise à l'échelle de l'architecture.

Il apparaît difficile d'imaginer des architectures fondamentalement différentes car les technologies sont optimisées pour ce type d'implémentation (avec notamment une forte présence de RAM dans les FPGA afin de faire du calcul centralisé). Dès lors tout projet industriel pragmatique se tournera vers ces solutions. Pourtant, alors que la limite de la loi de Moore s'approche, il est urgent de proposer des changements de paradigmes matériels plus drastiques. Dans ce cadre général, nous nous sommes plus particulièrement intéressés aux implémentations cellulaires des champs neuronaux dynamiques impulsionnels.

Les deux modèles proposés dans cette thèse reposent sur une transmission locale aléatoire de l'information qui traverse l'étendue de la grille de neurones. La transmission aléatoire permet de simuler la convolution à l'origine des interactions latérales des DNFs et essentielle à leur comportement émergent. Le modèle RSDNF de champs neuronaux dynamiques impulsionnels aléatoires, a pour principe de simuler le noyau de convolution gaussien des DNF classiques sans transmettre aucune indication sur les neurones émetteurs grâce à la décroissance exponentielle de la probabilité de transmission à longue distance des impulsions lorsque celles-ci sont transmises aléatoirement de proche en proche. L'étude approfondie de ce modèle valide ses performances comportementales et permet d'affiner la compréhension du rôle de ses différents paramètres. Son implantation matérielle montre néanmoins que l'objectif principal visé, la mise à l'échelle et la décentralisation des calculs, est atteint au détriment de la surface d'implantation malgré son

optimisation, et au prix d'une synchronisation coûteuse des phases de transmission. Le modèle CASAS-DNF de grille cellulaire de DNF impulsionnels aléatoires et asynchrones améliore le modèle précédent sur ces aspects grâce à la suppression des tampons d'impulsions obtenue au moyen d'une approche par flux de bits aléatoires des informations transmises entre neurones voisins. Si la surface d'implantation rejoint celle des approches centralisées, la vitesse de calcul s'en trouve pénalisée. En revanche, ce modèle s'avère compatible avec une implantation asynchrone, ce qui ouvre de nombreuses perspectives.

Ces contributions ont permis de développer les mécanismes principaux d'un calcul neuronal cellulaire. Grâce à la symétrie du noyau d'interactions latérales et à la définition des DNF impulsionnels, il a ainsi été possible de simplifier les communications latérales à un voisinage de quatre voisins avec deux bits d'informations (excitateur et inhibiteur). Il est intéressant de noter que la robustesse du comportement du DNF impulsionnel a permis de nombreuses optimisations et en permettrait probablement beaucoup d'autres qui n'ont pas été explorées. Nous n'avons pas simplement implémenté une équation de façon cellulaire dans un substrat numérique mais nous avons aussi testé les limites des mécanismes d'émergence des comportements attendus. Le bilan comparatif des différentes solutions envisagées illustre la variété des critères à prendre en compte, et montre qu'à ce stade, aucun des modèles et des méthodes d'implantation ne s'impose sur l'ensemble des aspects.

Les perspectives de ce travail sont nombreuses. Outre l'étude approfondie des variantes asynchrones ou fréquentielles du modèle CASAS, déjà évoquées dans le manuscrit, cette thèse nous amène à envisager des extensions de l'approche neuro-cellulaire dans un cadre qui pourra dépasser les champs neuronaux dynamiques simples, notamment au travers d'architectures multi-cartes permettant d'envisager des modèles cognitifs plus complexes, ou au travers de l'étude de versions cellulaires basées sur des solutions alternatives mettant en jeu une communication globale entre unités de calcul.

Architecture multi-carte

Les modèles que nous avons décrits sont suffisamment génériques pour s'adapter à une architecture multicarte. Les DNF peuvent être empilés topographiquement et les connections afférentes peuvent être implémentées de la même façon que les connections latérales : avec une couche de routage aléatoire. Le noyau afférent sera alors aussi exponentiel. Là encore les nombres aléatoires peuvent être partagés pour toute une pile de neurones de sorte qu'il ne soit pas nécessaire d'ajouter plus de générateurs aléatoires. On voit donc que cette architecture supporte facilement la mise à l'échelle en terme de nombre de cartes. Le temps de calcul reste constant alors que la surface augmente linéairement avec le nombre de couches et le nombre d'interconnexions entre ces couches. L'étude approfondie de ces perspectives nécessite néanmoins un travail très conséquent.

Convolution globale sur un substrat cellulaire

Puisque le point essentiel du calcul cellulaire proposé est la convolution globale par un noyau en forme de différence de gaussiennes on peut se demander quelles autres possibilités sont possibles pour faire ce calcul.

Impulsions simples latérales Les Lateral simple spikes, LSS en anglais, remplacent le routage aléatoire par un routage déterministe où les impulsions sont additionnées lorsqu'il y en

a plusieurs dans le même routeur [Rodriguez, 2015]. On peut alors déterminer la distance du neurone émetteur en se reposant sur une horloge globale.

Les avantages de ce modèle sont multiples. Il est déterministe donc une seule étape de diffusion est nécessaire. La surface des générateurs aléatoires est remplacée par celle des additionneurs nécessaires à chaque routeur. En fonction de la façon de générer les nombres aléatoires, cette surface sera plus ou moins avantageuse. Le noyau de convolution reste symétrique mais peut avoir une forme quelconque.

Le défaut principal est la synchronicité de l'architecture. Il est nécessaire d'avoir une horloge globale pour synchroniser tous les neurones.

Connectivité symétrique Un des principaux défauts des architectures présentées dans ce manuscrit est la complexité des routeurs par rapport à une architecture cellulaire idéale. Il serait souhaitable que la communication des informations soit symétrique afin de supprimer les routeurs, et ainsi augmenter substantiellement la robustesse aux fautes.

Une proposition pour résoudre ce problème suppose de changer les poids synaptiques à chaque itération de la propagation [Rodriguez, 2015]. Il est ainsi possible de compenser le poids synaptique en fonction du nombre de chemins possibles à cause des récurrences dues aux connexions symétriques.

Dans ce cas l'architecture est aussi complexifiée puisque de nombreux poids synaptiques doivent être stockés. Le défaut de synchronicité est toujours présent.

Automates cellulaires Au cours de mes recherches, j'ai proposé un automate cellulaire capable de reproduire la convolution globale mais avec un noyau linéaire.

Dans [Chappet de Vangel and Fix, 2016] il est montré que la dynamique émergente des DNF est possible avec un noyau linéaire. il est donc possible d'utiliser ce résultat pour implémenter DNF uni-dimensionnel basé sur cette convolution. Cet automate a beaucoup de propriétés intéressantes comme la symétrie et l'asynchronisme des neurones. Malheureusement je n'ai pas trouvé de solution similaire pour un DNF à deux dimensions (voir annexe B).

Réseaux de neurones cellulaires Les réseaux neuronaux cellulaires (CNN) introduits par Chua [Chua and Yang, 1988] sont plutôt destinés à des substrats VLSI qui présentent l'avantage de pouvoir encoder le potentiel membranaire de façon continue.

Dans [Giese, 2012] l'auteur postule que les CNN seraient un moyen efficace de calculer les DNF. Il fait pour ça le rapprochement avec les filtres de Gabor implémentés par [Shi, 1998]. En effet un noyaux gaussien est une instance d'un filtre de Gabor avec une fréquence centrale de 0.

La diffusion des courants de voisin en voisin se fait par l'intermédiaire de résistances et la valeur du potentiel membranaire latéral est codé par la tension d'un condensateur. On comprend donc que la diffusion du courant est limitée dans l'espace et limite la possibilité d'une convolution globale.

Cependant ce substrat est très prometteur pour implémenter des noyaux localisés utiles pour les mémoires de travail par exemple. De plus cette méthode pour calculer les potentiels latéraux a l'avantage d'être robuste aux fautes et aux perturbations. En effet la convolution émerge de propriétés physiques des composants et correspond à un équilibre stable.

Il y a donc encore de nombreuses voies de recherche pour améliorer la robustesse, l'efficacité énergétique et la compacité du substrat cellulaire proposé. Les propriétés des champs neuronaux

dynamiques seraient alors disponibles sur un substrat parallèle décentralisé et robuste permettant de nombreuses applications embarquées. Ce substrat peut aussi facilement s'intégrer dans une pile matérielle cellulaire où tous les composants interagissent de manière décentralisée. Il peuvent par exemple servir de couche décisionnelle pour des cartes d'apprentissage comme dans [Detorakis and Rougier, 2012], [Rodriguez, 2015], [Lefort et al., 2011] ou [Maggiani et al., 2016]. Ce substrat cellulaire est donc une brique simple et configurable permettant de construire des architectures plus complexes. Il reste cependant des défis conceptuels pour mettre au point des architectures génériques capables d'apprentissage et d'adaptation.

Ce manuscrit et plus généralement le champ de recherche des moyens de calculs non conventionnels cherchent à répondre aux problématiques actuelles des architectures centralisées. Le calcul cellulaire est un bon moyen de réunir la mémoire et les unités de calculs tout en conservant une architecture adaptée aux substrats bi-dimensionnels. Le type de calcul que nous pouvons proposer avec ce substrat n'est pas en mesure de remplacer le calcul conventionnel puisqu'il vise pour le moment principalement des applications en traitement d'image et en robotique. Mais ce domaine d'application est en pleine croissance, ce qui justifie déjà pleinement le développement d'architectures dédiées. A plus long terme, l'évolution conjointe des besoins applicatifs et des contraintes matérielles de plus en plus fortes pose de nombreux défis pour lequels l'approche neuro-cellulaire défendue dans cette thèse, et plus généralement les approches matérielles bio-inspirées, pourraient constituer un début d'élément de réponse.

Bibliographie

[Adelson and Bergen, 1985] Adelson, E. H. and Bergen, J. R. (1985). Spatiotemporal energy models for the perception of motion. *J. Opt. Soc. Am. A*, 2(2) :284–299.

[Alaghi, 2015] Alaghi, A. (2015). *The Logic of Random Pulses : Stochastic Computing*. PhD thesis, University of Michigan.

[Amari, 1977] Amari, S.-i. (1977). Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, 27 :77–87. 10.1007/BF00337259.

[Amit, 1992] Amit, D. J. (1992). *Modeling brain function : The world of attractor neural networks*. Cambridge University Press.

[Bade and Hutchings, 1994] Bade, S. L. and Hutchings, B. L. (1994). Fpga-based stochastic neural networks-implementation. In *FPGAs for Custom Computing Machines, 1994. Proceedings. IEEE Workshop on*, pages 189–198.

[Bastian et al., 2003] Bastian, A., Schöner, G., and Riehle, A. (2003). Preshaping and continuous evolution of motor cortical representations during movement preparation. *European Journal of Neuroscience*, 18(7) :2047–2058.

[Benda and Herz, 2003] Benda, J. and Herz, A. V. (2003). A universal model for spike-frequency adaptation. *Neural computation*, 15(11) :2523–2564.

[Benjamin et al., 2014] Benjamin, B., Gao, P., McQuinn, E., Choudhary, S., Chandrasekaran, A., Bussat, J.-M., Alvarez-Icaza, R., Arthur, J., Merolla, P., and Boahen, K. (2014). Neurogrid : A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, 102(5) :699–716.

[Bicho et al., 2010] Bicho, E., Louro, L., and Erlhagen, W. (2010). Integrating verbal and nonverbal communication in a dynamic neural field architecture for human-robot interaction. *Front Neurorobotics*, 4.

[Bressloff, 2012] Bressloff, P. C. (2012). Spatiotemporal dynamics of continuum neural fields. *Journal of Physics A : Mathematical and Theoretical*, 45(3) :033001.

[Canals et al., 2015] Canals, V., Alomar, M. L., Morro, A., Oliver, A., and Rossello, J. L. (2015). Noise-robust hardware implementation of neural networks. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

[Canals et al., 2016] Canals, V., Morro, A., Oliver, A., Alomar, M. L., and Rosselló, J. L. (2016). A new stochastic computing methodology for efficient neural network implementation. *IEEE Transactions on Neural Networks and Learning Systems*, 27(3) :551–564.

[Chappet de Vangel and Fix, 2016] Chappet de Vangel, B. and Fix, J. (2016). In the quest of efficient hardware implementations of dynamic neural fields : an experimental study on the influence of the kernel shape. In *2016 International Joint Conference on Neural Networks (IJCNN)*.

[Chappet de Vangel et al., 2014] Chappet de Vangel, B., Torres-Huitzil, C., and Girau, B. (2014). Spiking dynamic neural fields architectures on fpga. In *ReConFigurable Computing and FPGAs (ReConFig), 2014 International Conference on*, pages 1–6.

[Chappet de Vangel et al., 2015a] Chappet de Vangel, B., Torres-Huitzil, C., and Girau, B. (2015a). Randomly spiking dynamic neural fields. *ACM Journal of Emerging Technologies in Computing Systems*, 11(4) :1–26.

[Chappet de Vangel et al., 2015b] Chappet de Vangel, B., Torres-Huitzil, C., and Girau, B. (2015b). Stochastic and asynchronous spiking dynamic neural fields. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

[Chappet de Vangel et al., 2016] Chappet de Vangel, B., Torres-Huitzil, C., and Girau, B. (2016). Event based visual attention with dynamic neural field on fpga. In *Proceedings of the 10th International Conference on Distributed Smart Cameras*, ICDSC '16, New York, NY, USA. ACM.

[Cheung et al., 2016] Cheung, K., Schultz, S. R., and Luk, W. (2016). Neuroflow : A general purpose spiking neural network simulation platform using customizable processors. *Frontiers in Neuroscience*, 9(516).

[Chevallier and Tarroux, 2008] Chevallier, S. and Tarroux, P. (2008). Visual focus with spiking neurons. In *European Symposium on Artificial Neural Networks*, pages 385–389.

[Chua and Yang, 1988] Chua, L. O. and Yang, L. (1988). Cellular neural networks : Applications. *Circuits and Systems, IEEE Transactions on*, 35(10) :1273–1290.

[Clerc, 2012] Clerc, M. (2012). Standard particle swarm optimisation.

[Dalal et al., 2008] Dalal, I. L., Stefan, D., and Harwayne-Gidansky, J. (2008). Low discrepancy sequences for monte carlo simulations on reconfigurable platforms. In *2008 International Conference on Application-Specific Systems, Architectures and Processors*, pages 108–113. IEEE.

[Deneve et al., 2001] Deneve, S., Latham, P. E., and Pouget, A. (2001). Efficient computation and cue integration with noisy population codes. *Nature neuroscience*, 4(8) :826–831.

[Desimone, 1998] Desimone, R. (1998). Visual attention mediated by biased competition in extrastriate visual cortex. *Philosophical Transactions of the Royal Society of London B : Biological Sciences*, 353(1373) :1245–1255.

[Detorakis and Rougier, 2012] Detorakis, G. I. and Rougier, N. P. (2012). A neural field model of the somatosensory cortex : formation, maintenance and reorganization of ordered topographic maps. *PloS one*, 7(7) :e40257.

[Douglas and Martin, 2012] Douglas, R. J. and Martin, K. A. (2012). Behavioral architecture of the cortical sheet. *Current Biology*, 22(24) :R1033 – R1038.

[Engels and Schöner, 1995] Engels, C. and Schöner, G. (1995). Dynamic fields endow behavior-based robots with representations. *Robotics and Autonomous Systems*, 14(1) :55 – 77.

[Erlhagen and Bicho, 2006] Erlhagen, W. and Bicho, E. (2006). The dynamic neural field approach to cognitive robotics. *J Neural Eng*, 3(3) :R36–54.

[Erlhagen and Schöner, 2002] Erlhagen, W. and Schöner, G. (2002). Dynamic field theory of movement preparation. *Psychological Review*, 109(3) :545–572.

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

[Faisal et al., 2008] Faisal, A. A., Selen, L. P., and Wolpert, D. M. (2008). Noise in the nervous system. *Nature Reviews Neuroscience*, 9(4) :292–303.

[Fates, 2013] Fates, N. (2013). A guided tour of asynchronous cellular automata. In *International Workshop on Cellular Automata and Discrete Complex Systems*, pages 15–30. Springer.

[Faubel and Schöner, 2008] Faubel, C. and Schöner, G. (2008). Learning to recognize objects on the fly : a neurally based dynamic field approach. *Neural Networks Special Issue on Neuroscience and Robotics*, 21(4) :Pages 562–576.

[Fix, 2013] Fix, J. (2013). Template based black-box optimization of dynamic neural fields. *Neural Networks*, 46(0) :40 – 49.

[Fix et al., 2011a] Fix, J., Geist, M., Pietquin, O., and Frezza-Buet, H. (2011a). Dynamic neural field optimization using the unscented kalman filter. In *Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB), 2011 IEEE Symposium on*, pages 1–7.

[Fix et al., 2011b] Fix, J., Rougier, N. P., and Alexandre, F. (2011b). A dynamic neural field approach to the covert and overt deployment of spatial attention. *Cognitive Computation*, 3(1) :279–293.

[Fix et al., 2007] Fix, J., Vitay, J., and Rougier, N. P. (2007). A distributed computational model of spatial memory anticipation during a visual search task. In Butz, M. et al., editors, *Anticipatory Behavior in Adaptive Learning Systems : From Brains to Individual and Social Behavior*, volume LNAI 4520. Springer-Verlag Berlin Heidelberg.

[Fung and Amari, 2015] Fung, C. and Amari, S.-i. (2015). Spontaneous motion on two-dimensional continuous attractors. *Neural computation*, 27(3) :507–547.

[Gaines, 1967] Gaines, B. R. (1967). Stochastic computing. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 149–156. ACM.

[Giese, 2012] Giese, M. A. (2012). *Dynamic neural field theory for motion perception*, volume 469. Springer Science & Business Media.

[Girau, 1999] Girau, B. (1999). *Du parallelisme des modeles connexionnistes a leur implantation parallele*. PhD thesis. Thèse de doctorat dirigée par Cosnard, Michel Sciences et techniques École normale supérieure (Lyon) 1999.

[Girau and Vlassopoulos, 2011] Girau, B. and Vlassopoulos, N. (2011). Tiled cellular automata for area-efficient distributed random number generators. In *1st International conference on pervasive and embedded computing and communication systems - PECCS 2011*.

[Golomb and Amitai, 1997] Golomb, D. and Amitai, Y. (1997). Propagating neuronal discharges in neocortical slices : computational and experimental study. *Journal of neurophysiology*, 78(3) :1199–1211.

[Holi and Hwang, 1993] Holi, J. L. and Hwang, J.-N. (1993). Finite precision error analysis of neural network hardware implementations. *IEEE Transactions on Computers*, 42(3) :281–290.

[Hu and Zhang, 2010] Hu, X. and Zhang, B. (2010). A gaussian attractor network for memory and recognition with experience-dependent learning. *Neural computation*, 22(5) :1333–1357.

[Hubel and Wiesel, 1963] Hubel, D. and Wiesel, T. (1963). Shape and arrangement of columns in cat's striate cortex. *The Journal of physiology*, 165(3) :559–568.

[Hussein and Swift, 2015] Hussein, J. and Swift, G. (2015). *Mitigating Single-Event Upsets*. XILINX.

[Igel et al., 2001] Igel, C., Erlhagen, W., and Jancke, D. (2001). Optimization of dynamic neural fields. *Neurocomputing*, 36(1) :225–233.

[Indiveri, 2008] Indiveri, G. (2008). Neuromorphic vlsi models of selective attention : from single chip vision sensors to multi-chip systems. *Sensors*, 8(9) :5352–5375.

[Indiveri and Horiuchi, 2011] Indiveri, G. and Horiuchi, T. K. (2011). Frontiers in neuromorphic engineering. *Frontiers in Neuroscience*, 5.

[Indiveri et al., 2011] Indiveri, G., Linares-Barranco, B., Hamilton, T. J., Van Schaik, A., Etienne-Cummings, R., Delbruck, T., Liu, S.-C., Dudek, P., Häfliger, P., Renaud, S., et al. (2011). Neuromorphic silicon neuron circuits. *Frontiers in neuroscience*, 5.

[ITRS, 2014] ITRS (2014). Itrs2.0 white paper more moore. url=http ://www.itrs2.net/itrsreports.html. Accessed : 2016-15-30.

[Johnson et al., 2009] Johnson, J. S., Spencer, J. P., Luck, S. J., and Schoner, G. (2009). A Dynamic Neural Field Model of Visual Working Memory and Change Detection. *Psychological Science*, 20(5) :568–577.

[Joubert et al., 2012] Joubert, A., Belhadj, B., Temam, O., and Héliot, R. (2012). Hardware spiking neurons design : Analog or digital ? In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–5. IEEE.

[Kang et al., 2003] Kang, K., Shelley, M., and Sompolinsky, H. (2003). Mexican hats and pinwheels in visual cortex. *Proceedings of the National Academy of Sciences*, 100(5) :2848–2853.

[Koickal et al., 2007] Koickal, T. J., Hamilton, A., Tan, S. L., Covington, J. A., Gardner, J. W., and Pearce, T. C. (2007). Analog vlsi circuit implementation of an adaptive neuromorphic olfaction chip. *IEEE Transactions on Circuits and Systems I : Regular Papers*, 54(1) :60–73.

[Kopecz and Schöner, 1995] Kopecz, K. and Schöner, G. (1995). Saccadic motor planning by integrating visual information and pre-information on neural dynamic fields. *Biological cybernetics*, 73(1) :49–60.

[Lefort et al., 2011] Lefort, M., Boniface, Y., and Girau, B. (2011). Coupling bcm and neural fields for the emergence of self-organization consensus. In *From Brains to Systems*, pages 41–56. Springer.

[Lichtsteiner et al., 2008] Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128× 128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *IEEE journal of solid-state circuits*, 43(2) :566–576.

[London et al., 2010] London, M., Roth, A., Beeren, L., Häusser, M., and Latham, P. E. (2010). Sensitivity to perturbations in vivo implies high noise and suggests rate coding in cortex. *Nature*, 466(7302) :123–127.

[Madison and Nicoll, 1984] Madison, D. and Nicoll, R. (1984). Control of the repetitive discharge of rat ca 1 pyramidal neurones in vitro. *The Journal of Physiology*, 354(1) :319–331.

[Maggiani et al., 2016] Maggiani, L., Bourrasset, C., Quinton, J.-C., Berry, F., and Sérot, J. (2016). Bio-inspired heterogeneous architecture for real-time pedestrian detection applications. *Journal of Real-Time Image Processing*, pages 1–14.

[Maguire et al., 2007] Maguire, L., McGinnity, T., Glackin, B., Ghani, A., Belatreche, A., and Harkin, J. (2007). Challenges for large-scale implementations of spiking neural networks on FPGAs. *Neurocomputing*, 71(1-3) :13–29.

[Mahowald, 1992] Mahowald, M. (1992). *VLSI Analogs of Neuronal Visual Processing : A Synthesis of Form and Function*. PhD thesis, California Institute of Technology.

[Manohar, 2015] Manohar, R. (2015). Comparing stochastic and deterministic computing. *IEEE Computer Architecture Letters*, 14(2) :119–122.

[Massoud and Horiuchi, 2011] Massoud, T. and Horiuchi, T. (2011). A neuromorphic vlsi head direction cell system. *Circuits and Systems I : Regular Papers, IEEE Transactions on*, 58(1) :150–163.

[Mead, 1990] Mead, C. (1990). Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10) :1629–1636.

[Misra and Saha, 2010] Misra, J. and Saha, I. (2010). Artificial neural networks in hardware : A survey of two decades of progress. *Neurocomputing*, 74(1–3) :239 – 255. Artificial Brains.

[Mountcastle, 1957] Mountcastle, V. B. (1957). Modality and topographic properties of single neurons of cat's somatic sensory cortex. *Journal of neurophysiology*, 20(4) :408–434.

[Neil and Liu, 2014] Neil, D. and Liu, S.-C. (2014). Minitaur, an event-driven fpga-based spiking network accelerator. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 22(12) :2621–2628.

[Omondi and Rajapakse, 2006] Omondi, A. R. and Rajapakse, J. C. (2006). *FPGA implementations of neural networks*, volume 365. Springer.

[Palesi and Daneshtalab, 2014] Palesi, M. and Daneshtalab, M. (2014). *Routing algorithms in networks-on-chip*. Springer.

[Patel et al., 2007a] Patel, N., Nguang, S. K., and Coghill, G. (2007a). Neural network implementation using bit streams. *Neural Networks, IEEE Transactions on*, 18(5) :1488–1504.

[Patel et al., 2007b] Patel, N. D., Nguang, S. K., and Coghill, G. G. (2007b). Neural network implementation using bit streams. *IEEE Transactions on Neural Networks*, 18(5) :1488–1504.

[Piaget, 2013] Piaget, J. (2013). *The construction of reality in the child*, volume 82. Routledge.

[Pinto and Ermentrout, 2001] Pinto, D. J. and Ermentrout, G. B. (2001). Spatially structured activity in synaptically coupled neuronal networks : I. traveling fronts and pulses. *SIAM journal on Applied Mathematics*, 62(1) :206–225.

[Posner and Petersen, 1990] Posner, M. I. and Petersen, S. E. (1990). *Annual Review Neuroscience*, 13 :25–42.

[Potthast and Graben, 2009] Potthast, R. and Graben, P. B. (2009). Inverse problems in neural field theory. *SIAM Journal on Applied Dynamical Systems*, 8(4) :1405–1433.

[Quinton, 2010] Quinton, J.-C. (2010). Exploring and Optimizing Dynamic Neural Fields Parameters Using Genetic Algorithms. In *IEEE World Congress on Computational Intelligence 2010 - WCCI 2010*, Barcelona, Espagne.

[Rodriguez, 2015] Rodriguez, L. (2015). *Définition d'un substrat computationnel bio-inspiré : déclinaison de propriétés de plasticité cérébrale dans les architectures de traitement auto-adaptatif*. PhD thesis, Université de Cergy-Pontoise.

[Rostro-Gonzalez et al., 2010] Rostro-Gonzalez, H., Barron-Zambrano, J. H., Torres-Huitzil, C., and Girau, B. (2010). Low-cost hardware implementations for discrete-time spiking neural networks. In *Cinquième conférence plénière française de Neurosciences Computationnelles, "Neurocomp'10"*, Lyon, France.

[Rougier and Vitay, 2006] Rougier, N. P. and Vitay, J. (2006). Emergence of attention within a neural population. *Neural Networks*, 19(5) :573 – 581.

[Samsonovich and McNaughton, 1997] Samsonovich, A. and McNaughton, B. L. (1997). Path integration and cognitive mapping in a continuous attractor neural network model. *The Journal of neuroscience*, 17(15) :5900–5920.

[Sandamirskaya, 2013] Sandamirskaya, Y. (2013). Dynamic neural fields as a step toward cognitive neuromorphic architectures. *Frontiers in Neuroscience*, 7.

[Sandamirskaya and Schoner, 2008] Sandamirskaya, Y. and Schoner, G. (2008). Dynamic field theory of sequential action : A model and its implementation on an embodied agent. In *Development and Learning, 2008. ICDL 2008. 7th IEEE International Conference on*, pages 133–138.

[Sarpeshkar et al., 1998] Sarpeshkar, R., Lyon, R. F., and Mead, C. (1998). A low-power wide-dynamic-range analog vlsi cochlea. In *Neuromorphic systems engineering*, pages 49–103. Springer.

[Schemmel et al., 2010] Schemmel, J., Bruderle, D., Grubl, A., Hock, M., Meier, K., and Millner, S. (2010). A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 1947–1950. IEEE.

[Schöner et al., 1995] Schöner, G., Dose, M., and Engels, C. (1995). Dynamics of behavior : Theory and applications for autonomous robot architectures. *Robotics and Autonomous Systems*, 16(2–4) :213 – 245. Moving the Frontiers between Robotics and Biology.

[Shi, 1996] Shi, B. E. (1996). Spatio-temporal image filtering with cellular neural networks. In *Neural Networks, 1996., IEEE International Conference on*, volume 3, pages 1410–1415 vol.3.

[Shi, 1998] Shi, B. E. (1998). Gabor-type filtering in space and time with cellular neural networks. *IEEE Transactions on Circuits and Systems I : Fundamental Theory and Applications*, 45(2) :121–132.

[Simmering et al., 2008] Simmering, V. R., Schutte, A. R., and Spencer, J. P. (2008). Generalizing the dynamic field theory of spatial cognition across real and developmental time scales. *Brain Research*, 1202(0) :68 – 86. Computational Cognitive Neuroscience.

[Sipper, 1999] Sipper, M. (1999). The emergence of cellular computing. *IEEE Computer*, 32(7) :18–26.

[Swindale, 1996] Swindale, N. (1996). The development of topography in the visual cortex : a review of models. *Network : Computation in neural systems*, 7(2) :161–247.

[Taouali et al., 2010] Taouali, W., Viéville, T., Rougier, N. P., and Alexandre, F. (2010). On asynchronous dynamic neural field computation. In *Cinquième conférence plénière française de Neurosciences Computationnelles, "Neurocomp'10"*, Lyon, France.

[Taube and Bassett, 2003] Taube, J. S. and Bassett, J. P. (2003). Persistent neural activity in head direction cells. *Cerebral Cortex*, 13(11) :1162–1172.

[Taylor, 1999] Taylor, J. G. (1999). Neural 'bubble' dynamics in two dimensions : foundations. *Biological Cybernetics*, 80 :393–409. 10.1007/s004220050534.

[Thelen et al., 2001] Thelen, E., Schöner, G., Scheier, C., and Smith, L. B. (2001). The dynamics of embodiment : A field theory of infant perseverative reaching. *Behavioral and brain sciences*, 24(01) :1–34.

[Vainbrand and Ginosar, 2011] Vainbrand, D. and Ginosar, R. (2011). Scalable network-on-chip architecture for configurable neural networks. *Microprocessors and Microsystems*, 35(2) :152–166.

[Vazquez et al., 2011] Vazquez, R., Girau, B., and Quinton, J. (2011). Visual attention using spiking neural maps. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2164 –2171.

[Vitay et al., 2005] Vitay, J., Rougier, N. P., and Alexandre, F. (2005). A distributed model of spatial visual attention. In S. Wermter, G. P. and Elshaw, M., editors, *Biomimetic Neural Learning for Intelligent Robotics*, volume 3575 of *Lecture Notes in Computer Science*, pages 54–72. Springer-Verlag.

[Vlassopoulos and Girau, 2014] Vlassopoulos, N. and Girau, B. (2014). A Metric for Evolving 2-D Cellular Automata As Pseudo-Random Number Generators. *Journal of Cellular Automata*.

[Wang et al., 2004] Wang, X.-J., Tegnér, J., Constantinidis, C., and Goldman-Rakic, P. (2004). Division of labor among distinct subtypes of inhibitory neurons in a cortical microcircuit of working memory. *Proceedings of the National Academy of Sciences of the United States of America*, 101(5) :1368–1373.

[Wilimzig et al., 2006] Wilimzig, C., Schneider, S., and Schöner, G. (2006). The time course of saccadic decision making : Dynamic field theory. *Neural Networks*, 19 :1059–1074.

[Wilson and Cowan, 1972] Wilson, H. R. and Cowan, J. D. (1972). Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical Journal*, 12(1) :1 – 24.

[Wilson and Cowan, 1973] Wilson, H. R. and Cowan, J. D. (1973). A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue. *Biological Cybernetics*, 13 :55–80. 10.1007/BF00288786.

[Wimmer et al., 2014] Wimmer, K., Nykamp, D. Q., Constantinidis, C., and Compte, A. (2014). Bump attractor dynamics in prefrontal cortex explains behavioral precision in spatial working memory. *Nature neuroscience*, 17(3) :431–439.

[Xie and Giese, 2002] Xie, X. and Giese, M. A. (2002). Nonlinear dynamics of direction-selective recurrent neural media. *Physical Review E*, 65(5) :051904.

[Xilinx, 2014] Xilinx (2014). *7 Series FPGAs Configurable Logic Block, User Guide*. XILINX.

[y Cajal, 1888] y Cajal, S. R. (1888). *Estructura de los centros nerviosos de las aves*.

[Zhang, 1996] Zhang, K. (1996). Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble : a theory. *The journal of neuroscience*, 16(6) :2112–2126.

[Zibner et al., 2010] Zibner, S. K. U., Faubel, C., Iossifidis, I., and Gregor, S. (2010). Scene representation based on dynamic field theory : From human to machine. *Frontiers in Computational Neuroscience*, page 5.

Annexe A

Optimisation des paramètres des modèles

A.1 Introduction

Tout au long du manuscrit nous avons utilisé l'optimisation méta-heuristique des essaims particulaires afin de trouver les paramètres pour les modèles que nous avons développés et implémentés.

Le problème d'optimisation qui cherche à trouver les paramètres d'un modèle est connu comme une forme de problème inverse. Il est en général très difficile d'être certain d'obtenir la meilleure solution. La performance d'une solution va donc être estimée à l'aide d'une fonction réelle qu'il va falloir minimiser ou maximiser. Etant donné que l'on peut facilement ramener un problème de maximisation à un problème de minimisation, on considère ici des problèmes de minimisation d'une fonction objectif f.

Il y a deux types d'optimisation : l'optimisation locale et l'optimisation globale. L'optimisation locale utilise par exemple une descente de gradient (Gauss-Newton) ou la recherche directe (simplexe). Son principal défaut est que le résultat dépend principalement de la position initiale de la recherche et que la solution sera souvent bloquée dans un minimum local. Afin de résoudre ce problème, l'optimisation globale utilise des moyens aléatoires pour explorer l'espace des solutions et sortir des minima locaux. Pour ces approches les algorithmes évolutifs (les algorithmes génétiques, les stratégies évolutives basées sur l'adaptation de matrice de covariance CMA-ES, les essaims particulaires et autres) sont inspirés des mécanismes d'optimisation émergents dans la nature. Ces algorithmes ont eu beaucoup de succès de par leur simplicité et leurs performances.

Les méta-heuristiques ont déjà été utilisées pour trouver les paramètres de modèles plus ou moins complexes basés sur des DNF. Dans [Quinton, 2010] l'auteur utilise un algorithme génétique afin de trouver les paramètres pour des scénarios de suivi de cible en deux dimensions. Dans [Fix, 2013] l'auteur compare les essaims particulaires et CMA-ES avec des scénarios de suivi de cible et de mémoire de travail en une dimension. [Igel et al., 2001] propose une méthode combinant une approche de descente de gradient et CMA-ES. D'autres méthodes ont été proposées en utilisant des filtres particulaires [Fix et al., 2011a] ou une approche plus formelle basée sur la pseudo inverse de Moose-Penrose [Potthast and Graben, 2009].

Un comparatif de ces méthodes n'est pas l'objectif de cette annexe et n'ayant pas une connaissance exhaustive de toutes les méthodes possibles nous avons choisi les essaims particulaires pour leur simplicité et leur efficacité.

A.2 Algorithme

L'algorithme des essaims particulaires est inspiré de la dynamique de collaboration entre des animaux tels que les groupes d'oiseaux. Des interactions simples entre particules permettent de faire émerger une solution à un problème donné. La position d'une particule dans l'espace des paramètres représente une solution du problème. A cette solution va être associée une performance qui est le résultat de la fonction de coût. La vitesse de la particule change en fonction de son voisinage (coopération) et de sa meilleure position et de son inertie. Et ainsi sa nouvelle position est calculée.

Nous utilisons ici un algorithme appelé SPSO2006 proposé par [Clerc, 2012]. Cet algorithme est un algorithme d'essaims particulaires standard avec une topologie aléatoire et adaptative.

A.2.1 Topologie adaptative

La topologie détermine les interactions entre les particules. Le paramètre k détermine le nombre moyen de voisins tirés aléatoirement dans le reste de la population. Dans notre implémentation le nombre de voisins est tiré dans une distribution gaussienne de moyenne k et de variance 0.5.

Le point important de cet algorithme est que la topologie change dès qu'il n'y a plus d'amélioration de la meilleure performance.

A.2.2 Mise à jour des positions

Soit d le nombre de paramètres du modèle. On note $v_i \in \mathbb{R}^d$ la vélocité de la particule i, $x_i \in \mathbb{R}^d$ sa position, $p_i \in \mathbb{R}^d$ sa meilleure position et $l_i \in \mathbb{R}^d$ la meilleure des meilleures positions de ses voisins.

La mise à jour de la particule du temps t au temps $t+1$ est donnée par

$$v_i^{t+1} = \Omega v_i^t + r_p \phi_p (p_i^t - x_i^t) + r_g \phi_g (l_i^t - x_i^t) \tag{A.1}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{A.2}$$

où Ω, ϕ_p et ϕ_g sont des constantes de l'algorithme (on utilise $\Omega = 0.721$ et $\phi_p = \phi_g = 1.193$), r_p et r_g sont des réels tirés dans une distribution uniforme entre 0 et 1.

A.2.3 Confinement

Le confinement de la position et de la vélocité des particules permet de rester dans les bornes d'optimisation imposées par le problème.

Nous avons directement utilisé la méthode de confinement proposée par [Clerc, 2012].

A.2.4 Arrêt de l'optimisation

En général l'optimisation s'arrête quand la meilleure performance est plus petite qu'une performance acceptable ou quand le nombre limite de mises à jour de toute la population (ou

époque) est atteint. L'algorithme utilisé dans cette thèse est le suivant :

Data: N nombre de particules, Ω ,ϕ_p, ϕ_g. $acceptableFitness$
$x \leftarrow initPopulation(N, boundsPosition)$
$p \leftarrow copy(x)$
$v \leftarrow initVelocities(N, boundsVelocity)$
Calculer la topologie
/ Initialise le fitness et trouve la meilleure particule. */*
for $i \in [0, N-1]$ **do**
  $fitness[i] \leftarrow calculerFitness(x[i])$
end
$bestFitness \leftarrow min(fitness)$
$previousBestFitness \leftarrow bestFitness$
$bestPart \leftarrow argmin(fitness)$
/ Mise à jour des particules dans un ordre aléatoire.*/*
while $bestFitness > acceptableFitness$ **do**
  / Choisit une particule. */*
  $i \leftarrow random(0, N-1)$
  $l[i] \leftarrow getBestNeighbor(i)$
  Mettre à jour les vélocités de la particule / selon équation A.1 */*
  Mettre à jour la position de la particule / selon équation A.2 */*
  Confinement
  / Calcule le nouveau fitness. */*
  $fitness[i] \leftarrow calculerFitness(x[i])$
  $bestFitness \leftarrow min(fitness)$
  $bestPart \leftarrow argmin(fitness)$
  if $bestFitness = previousBestFitness$ **then**
    Calculer la topologie / nouveau tirage aléatoire des voisins */*
  else
    $previousBestFitness \leftarrow bestFitness$
  end
end

A.3 Fonction objectif pour les DNF

Il est souvent souhaitable d'utiliser plusieurs scénarios lors du calcul de la fonction objectif. Par exemple lors de l'optimisation pour un DNF de suivi de cible on utilisera de préférence un scénario de suivi simple et un scénario avec bruit et distractions.

La performance totale est alors une moyenne des performances de chaque scénario. La performance d'un scénario est calculée à partir de deux caractéristiques disponibles à chaque itération de la simulation : l'erreur de distance E et le nombre d'activations en dehors du cluster A (voir section 1.2.1.3.2 pour plus de détails).

La performance f sera alors calculée avec

$$f = 10 < E > + < A > \tag{A.3}$$

où <> est la moyenne tout au long de la simulation. Pour les DNF impulsionels, le coeficient de l'erreur est augmanté à 100.

A.4 Étude de cas : optimisation du noyau DoE

Pour les DNF plusieurs détails sont à prendre en compte afin de faciliter la convergence. On souhaite optimiser les paramètres suivants : k_e, k_i, w_e, w_i, h et τ. Quel que soit le scénario cible, la dynamique des DNF émerge si $k_e > k_i$ et $w_e < w_i$. C'est a dire si on a une coopération locale et une compétition globale.

Il est possible d'imposer ces conditions lors de l'optimisation en transformant l'espace des paramètres optimisés. Ainsi on pose $K = \frac{k_i}{k_e}$ et $W = \frac{w_e}{w_i}$. Les contraintes sur K et W assurent que les conditions énoncées ci-dessus sont valides : $K \in [0, 1]$ et $W \in [0, 1]$.

A.4.1 Suivi de cible

Pour le suivi de cible robuste il n'est pas nécessaire d'optimiser h, on fixe donc $h = 0$. On peut ensuite fixer $w_i = 1$ afin de s'assurer d'une inhibition globale. Les paramètres restant sont K, W, $\tau \in [0, 1]$ et $k_e \in [0, 5]$.

Deux scénarios sont utilisés un avec du suivi sans bruit ni distractions et un avec 5 distractions et un bruit gaussien de variance 1.

A.4.2 Mémoire de travail

Pour la mémoire de travail il faut optimiser $h \in [-1, 0]$ et $w_i \in [0, 1]$. En fonction de la tâche ciblée et notamment de la taille des stimuli que l'on souhaite garder en mémoire, il faut faire attention à normaliser l'espace des variables optimisées. Dans le cas de la mémoire de travail, la taille du noyau est très petite avec des poids synaptiques de l'ordre de $10\times^{-3}$. Par conséquent il faut normaliser l'espace en posant $w'_i = w_i \times 10^3$ par exemple, en gardant la contrainte $w'_i \in [0, 1]$.

Deux scénarios de mémoire de travail avec suivi sont utilisés, un avec 3 distractions et un bruit de variance 0.1 et l'autre avec 5 distractions et un bruit de variance 0.3.

Annexe B

Automate cellulaire pour les champs neuronaux dynamiques

La conception de ce modèle découle d'une prise de conscience des défauts inhérents aux modèles RSDNF et CASAS et plus particulièrement à leur mode de routage XY. Comme nous l'avons vu dans les chapitres précédents, le routage XY introduit une asymétrie qui est difficile à compenser. Dans le modèle RSDNF, l'asymétrie est responsable de retards importants dans le routage ce qui a pour conséquence une grande complexité temporelle et une asymétrie dans la propagation des impulsions. Cette asymétrie rend obligatoire la séparation des phases de calcul neuronal et des phases de propagation des impulsions limitant l'asynchronisme et la décentralisation du modèle. Une partie de ces défauts ont été corrigés par le modèle CASAS mais là encore, l'asymétrie oblige à choisir un flux de bits très long augmentant ainsi la complexité temporelle.

La solution que nous proposons d'étudier dans ce chapitre s'inspire plus profondément des paradigmes du calcul cellulaire. Notamment du fait que les substrats naturels (qui sont robustes) sont fondamentalement symétriques. Les lois de la physique sont symétriques. Il est donc raisonnable de s'en inspirer afin d'obtenir un substrat de calcul plus robuste. La diffusion de la matière définie par la loi de Fick ne permet pas de simuler la convolution globale à l'origine des interactions latérales des DNF. Il existe cependant des automates cellulaires assez simple permettant de réaliser une diffusion linéaire des états, nous allons nous en inspirer pour ce modèle.

B.1 Cas d'une seule activation

Nous allons montrer ici l'idée générale de cette implémentation en se plaçant dans le contexte simplifié d'une seule activation. On considère les même neurones impulsionnels que nous avons utilisés jusqu'à présent en utilisant un pas de discrétisation temporelle dt suffisamment petit pour assurer qu'il n'y a qu'une seule activation par période de mise à jour des neurones.

La diffusion de cette activation dans toute la carte se fait alors avec un automate cellulaire homogène de diffusion. Comme précédemment il y aura une couche de diffusion pour les informations excitatrices et une couche de diffusion pour les informations inhibitrices. Nous allons définir la règle de mise à jour de l'automate cellulaire correspondant à une de ces deux couches.

La valeur discrète de chaque cellule est noté $\gamma \in [0, \alpha]$ où $\alpha \in \mathbb{N}$ est la valeur d'activation maximale.

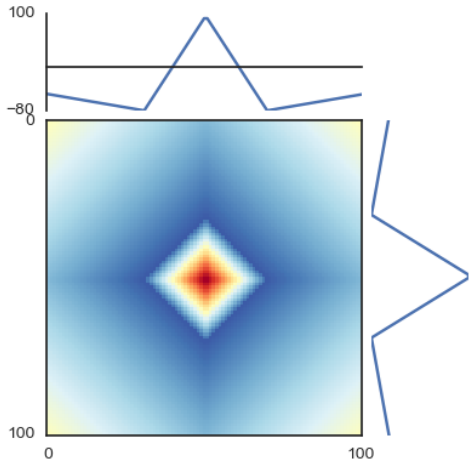


FIGURE B.1 – Noyau d'interactions latérales émergent de la diffusion d'une activation par deux couches d'un automate cellulaire de diffusion ($\alpha_e = 200, \beta_e = 10, \alpha_i = 100, \beta_i = 1$).

La règle locale de mise à jour dépend des ν voisins de la cellule et est définie comme suit :

$$\gamma^{t+1} = \begin{cases} max_\nu \gamma_\nu^t - \beta \text{ si } max_\nu \gamma_\nu^t > \gamma^t \\ \gamma^t \text{ sinon.} \end{cases} \tag{B.1}$$

où $\beta \in \mathbb{N}$ est le coefficient de décroissance linéaire. Le temps de diffusion de l'automate est de $2R$ dans le cas d'un voisinage de von Neumann (résolution $R \times R$). Après la diffusion les potentiels neuronaux sont mis à jour et les potentiels synaptiques remis à zéro. Si il y a activation la cellule sous-jacente prend la valeur maximale $\gamma = \alpha$.

On peut voir sur la figure B.1 le noyaux d'interactions latérales de deux couches de cette automate, une couche excitatrice ($\alpha_e = 200, \beta_e = 10$) et une couche inhibitrice ($\alpha_i = 100, \beta_i = 1$).

Le noyau d'interactions latérales est donc :

$$w(d) = max(0, \alpha_e - \beta_e d) - max(0, \alpha_i - \beta_i d) \tag{B.2}$$

B.2 Avantages d'une couche de diffusion cellulaire

Les avantages de ce type de diffusion sont multiples. La surface d'implémentation est raisonnable puisqu'on économise les 8 routeurs de chaque cellule dans CASAS ainsi que les générateurs de nombres pseudo-aléatoires.

La vitesse de l'architecture est avantageuse vu que le temps de diffusion est de $2R$ uniquement et le comportement est plus stable vu qu'il est déterministe.

Un des aspects les plus attrayants des automates cellulaires de diffusion est leur robustesse aux fautes permanentes ou temporaires. Cette robustesse est due aux propriétés mêmes de la diffusion qui permet de contourner n'importe quel obstacle. Cette propriété est utile si par exemple toute une zone de la carte est désactivée comme on peut le voir sur la figure B.2.

Une propagation des interactions latérales avec un automate cellulaire de diffusion linéaire paraît donc attrayante. Cependant, avant d'aller plus loin il faut s'assurer que ce type de noyau linéaire est compatible avec la dynamique des DNFs.

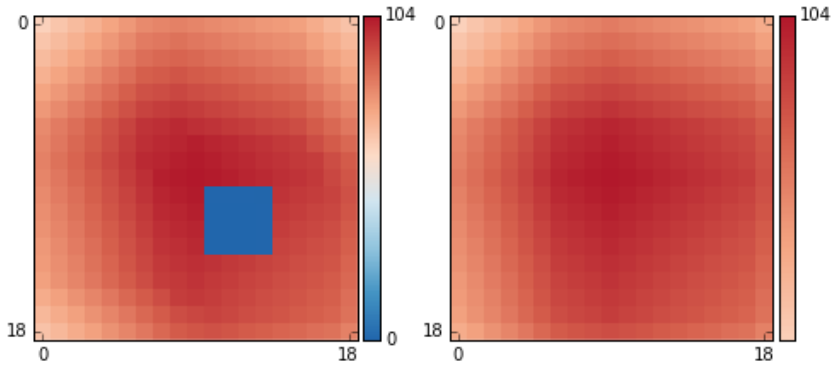


FIGURE B.2 – Diffusion sur une couche cellulaire dans le cas où une zone est complètement désactivée (à gauche en bleu), diffusion normale à droite.

B.3 Étude du noyau d'interactions latérales linéaire

Il est possible d'optimiser un DNF avec un noyau d'interaction de type linéaire comme dans [Chappet de Vangel and Fix, 2016]. Cette étude ne montre pas de restrictions particulière pour un noyau linéaire car celui-ci respecte les conditions énoncées par Amari pour la stabilité d'un noyau. Nous utilisons un algorithme d'optimisation en essaim de particules afin de trouver le meilleur jeu de paramètres pour trois scenarios : robustesse, bruit et changement de cible. En utilisant les même paramètres que précédemment le meilleur individu trouvé après 800 évaluations est celui de la table B.1.

TABLE B.1 – Paramètres obtenus par optimisation par essaim de particules sur trois scénarios. Les paramètres sont normalisés pour un espace de dimension 1.

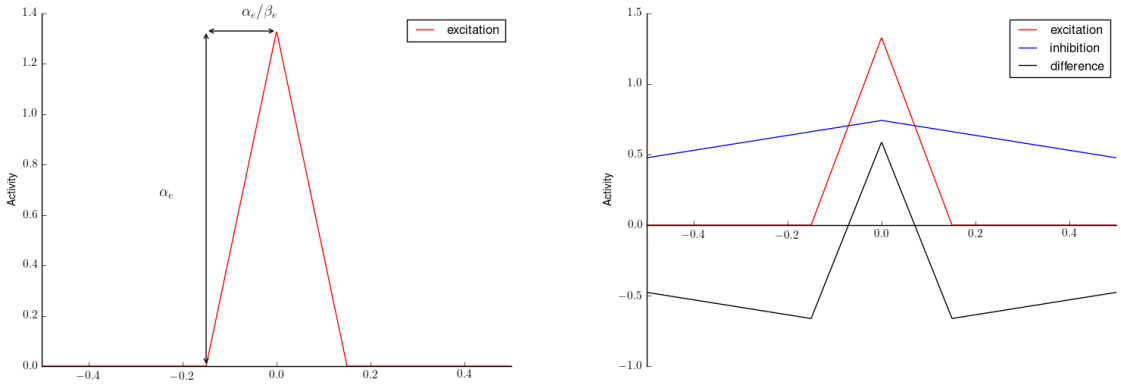
α_e	α_i	β_e	β_i
1.86	1.55	3.53	1.49

Il est donc possible de simuler le comportement du DNF discret avec ce noyau latéral en utilisant les paramètres obtenus. Les tests préliminaires montre une baisse de la performance sur la robustesse aux distractions. Cependant le scénario de suivi est validé (voir figure B.3).

B.4 Cas de plusieurs activations

Diminuer dt afin d'avoir au plus une seule activation par itération est très coûteux car il faut diviser dt par au moins 100 ou plus en fonction de la résolution et de τ. Il serait donc plus intéressant d'avoir un automate cellulaire capable de prendre en compte plusieurs activations. Nous allons montrer ici les principes généraux d'un tel automate bien que la solution complète n'ait pas été trouvée.

Dans cette version, γ prend les valeurs discrètes dans l'espace $[0, \alpha P]$ où P est le nombre maximal d'activations attendu. Pour montrer les défis posés par cet automate nous utiliserons un automate uni-dimensionnel.



(a) (b)

FIGURE B.3 – Fonction des poids latéraux obtenue avec l'automate cellulaire de diffusion. (a) la forme de la contribution excitatrice est déterminée par les paramètres α_a et β_b. En utilisant une deuxième couche inhibitrice on obtient une forme proche de la différence de gaussiennes (b).

On considère un voisinage de rayon 1 et plus spécifiquement le fait que le potentiel γ_i d'un voisin soit supérieur à celui de la cellule courante γ. Nous appelons s_i la variable booléenne : $s_i = (\gamma_i > \delta + \beta)$, $i \in [0, 1]$ pour les voisins est et ouest. On considère trois cas différents :

1. Deux voisins opposés sont supérieurs. $collision = s_0 \wedge s_1$. C'est une condition de collision.
2. Il y a un voisin supérieur mais il n'y a pas collision. $diffusion = (s_0 \vee s_1) \wedge (\overline{collision})$. C'est la condition de diffusion.

On défini alors la règle de mise à jour de l'automate avec :

$$\gamma^{t+1} = \begin{cases} \gamma_{i_0}^t + \gamma_{i_1}^t - 2\beta \text{ si } collision \\ max_{i=0,1}\gamma_i^t - \beta \text{ si } diffusion \\ \delta^t \text{ sinon} \end{cases} \tag{B.3}$$

L'initialisation est faite à $t = 0$ avec $\gamma^0 = \alpha$ si le neurone est activé.

B.4.1 Comportement

La figure B.5 montre un exemple du type de diffusion possible avec un tel automate.

Même s'il y a une petite différence, il semble donc possible d'intégrer plusieurs activations en même temps. Cependant cette règle très simple de mise à jour ne marche pas à chaque fois car elle ne détecte pas tous les types de collisions. De même pour un automate bi-dimensionnel la détection de collision de vagues de diffusion est un problème difficile à résoudre.

Ce modèle de diffusion des informations latérales de façon linéaire est prometteur pour ses bonnes propriétés de robustesse, sa surface restreinte grâce à l'abandon des routeurs et sa vitesse de diffusion compétitive. Nous avons montré qu'un noyau de type différence de fonctions

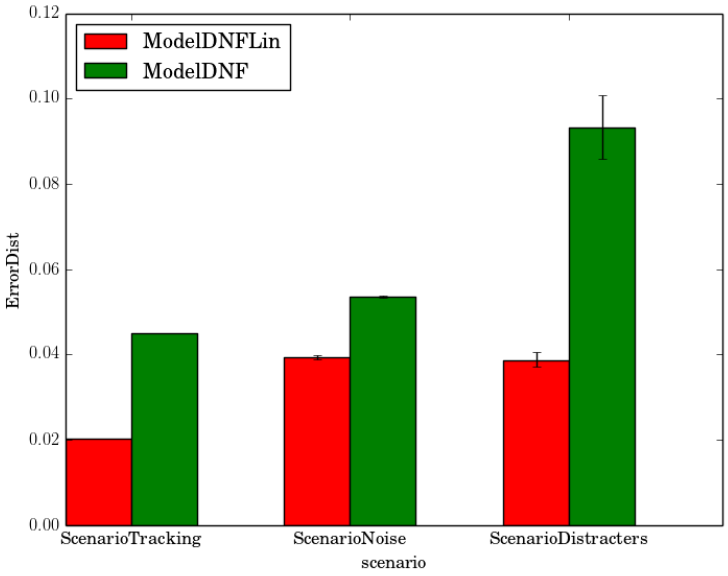


FIGURE B.4 – Performance du DNF avec un noyau linéaire (DNFLin) comparé au noyau standard.

linéaires pouvait conserver la dynamique des DNFs. Cependant la réalisation de l'automate cellulaire chargé de la diffusion des potentiels latéraux nécessite plus de travail afin de résoudre les problèmes de collisions. Une future étude devrait aussi étudier la capacité asynchrone de ce type de modèle qui rendrait la règle de mise à jour encore plus complexe.

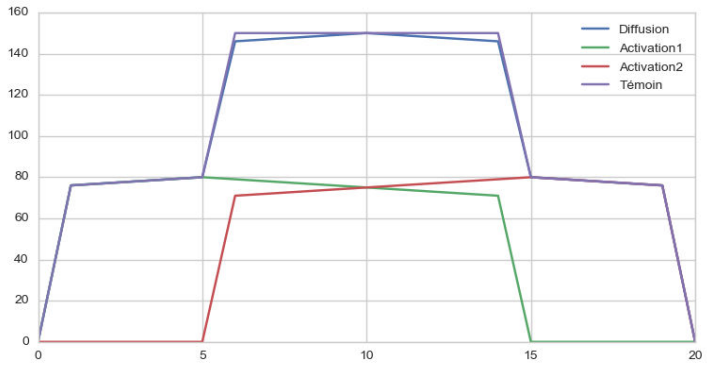


FIGURE B.5 – Diffusion sur une couche de l'automate cellulaire uni-dimensionnel dans le cas de deux activations. Les courbes rouge et verte représentent les contributions de ces deux activations. La courbe violette représente une diffusion idéale des informations d'activations et la courbe bleue le résultat de la diffusion avec l'automate cellulaire de diffusion.

Annexe C

CASAS-DNF : mise à l'échelle des performances

Il est possible de trouver les paramètres du modèle CASAS-DNF de façon formelle afin de voir comment la résolution peut les affecter. Nous allons reprendre les résultats du chapitre étudiant le modèle RSDNF sur le nombre de prédécesseurs de chaque routeur illustré sur la figure C.1.

Les routeurs verticaux ont comme prédécesseurs tous les neurones en dessous (pour les routeurs nord) ou au-dessus (pour les routeurs sud) du routeur. A la sortie d'un cluster d'activation de P neurones il y a donc $\sqrt{P}$ prédécesseurs aux routeurs verticaux au dessus et en dessous du cluster. Le nombre de prédécesseurs des routeurs horizontaux est beaucoup plus important : tous les routeurs à droite (pour le routeur est) ou à gauche (pour le router ouest). Par conséquent les P neurones du cluster d'activation sont prédécesseurs d'un routeur est à droite du cluster et ouest à gauche du cluster.

Nous allons utiliser les valeurs P_{min} et P_{max} pour évaluer les paramètres. Pour cela on déduit l'intervalle des nombres à encoder de façon à déterminer les paramètres N et δ de la diffusion. Le nombre maximal est celui à la sortie des routeurs horizontaux à droite et à gauche des clusters :

$$a_{max} = P_{max}\delta, \tag{C.1}$$

en négligeant les multiplications par les poids synaptiques. Le nombre minimal à encoder se situe en fin de propagation, c'est à dire à distance maximale du cluster que l'on estimera à $2R$ et pour les routeurs qui subissent le moins d'additions, c'est à dire les routeurs horizontaux.

$$a_{min} = (\sqrt{P_{min}}\delta)w^{2R} \tag{C.2}$$

pour une couche de propagation avec le poids synaptique $w = w_e$ (couche excitatrice) ou $w = w_i$ pour la couche inhibitrice.

On fait l'hypothèse que l'addition stochastique utilisant la porte OU est correcte pour les nombres $a < 0.1$ pour lesquels l'erreur engendrée sera inférieure à 0.01. Cela donne donc la borne positive :

$$a_{max} = P_{max}\delta = 0.1 \iff \delta = 0.1/P_{max} \tag{C.3}$$

Le plus petit nombre qu'un flux de bits de taille N permet d'encoder est $1/N$. La borne négative nous donne N :

$$1/N = a_{min} = (\sqrt{P_{min}}\delta)w^{2R} \iff N = \frac{1}{(\sqrt{P_{min}}\delta)w^{2R}} \tag{C.4}$$

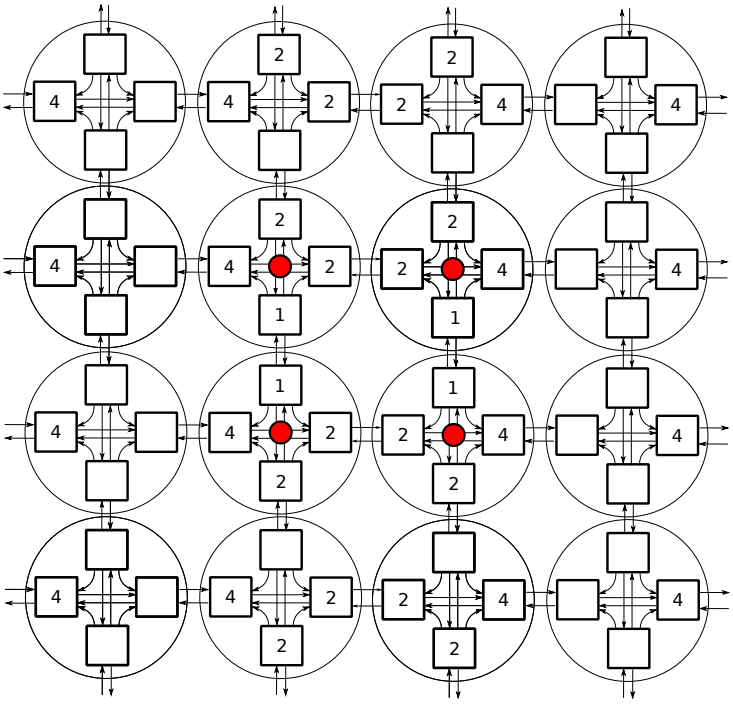


FIGURE C.1 – Nombre de neurones activés parmi les prédécesseurs pour chaque routeur.

Il faut aussi prendre en compte la mise à l'échelle des poids latéraux qui permet de conserver une forme de différence d'exponentielles constante quel que soit la résolution :

$$w = w'^{1/R} \tag{C.5}$$

La version normalisée de l'équation C.6 est donc :

$$N = \frac{1}{(\sqrt{P_{min}}\delta)w'^2} \tag{C.6}$$

Par exemple avec $R = 50$ dans un scénario de suivi de cible avec les paramètres suivants pour un noyaux DoE $ke = 0.46$, $ki = 0.41$, $we = 0.11$, $wi = 0.42$, 50 simulations donnent $P_{min} = 5$ et $P_{max} = 45$.

On obtient donc $\delta = 0.002$, $N_e \approx 16066$ et $N_i \approx 1102$. Il y a donc un très grand écart entre les besoins de précision pour la couche excitatrice et la couche inhibitrice. Ceci est dû aux très petites valeurs de l'excitation en dehors du noyau d'excitation lui-même. Vu qu'elles sont négligeables nous allons limiter la précision des poids excitateurs à celle des poids inhibiteurs.

On peut donc définir le nombre d'itérations nécessaires à la phase de diffusion comme le nombre d'itérations pour que le flux de bits de taille N atteigne tous les neurones de la carte $t_d = N + 2R$. Dans notre exemple on a donc $td = 1102 + 100 = 1202$. Pour $R = 100$, $N = 2071$ donc $td = 2271$.

Ces temps de diffusions semblent raisonnables mais ne correspondent pas à la réalité expérimentale. La raison principale est que cette analyse ne prend pas en compte la variabilité de

la transmission. Si la longueur du flux de bit est choisie pour pouvoir encoder δ par exemple, cela veut dire qu'en moyenne la valeur de δ sera correcte mais on n'a aucune certitude sur la variabilité. Il faut donc faire une analyse probabiliste de cette variabilité afin de connaître la longueur souhaitable pour le flux de bit pour restreindre la variance dans un certain intervalle.

Résumé

Dans la recherche permanente de solutions pour dépasser les limitations de plus en plus visibles de nos architectures matérielles, le calcul non-conventionnel offre des alternatives variées comme l'ingénierie neuromorphique et le calcul cellulaire. Comme von Neumann qui s'était initialement inspiré du cerveau pour concevoir l'architecture des ordinateurs, l'ingénierie neuromorphique prend la même inspiration en utilisant un substrat analogique plus proche des neurones et des synapses. Le calcul cellulaire s'inspire lui des substrats de calcul naturels (chimique, physiques ou biologiques) qui imposent une certaine localité des calculs de laquelle va émerger une organisation et des calculs.

La recherche sur les mécanismes neuronaux permet de comprendre les grands principes de calculs émergent des neurones. Un des grand principes que nous allons utiliser dans cette thèse est la dynamique d'attracteurs d'abord décrite par Amari (champs neuronaux dynamiques, ou DNF pour dynamic neural fields), Amit et Zhang (réseaux de neurones à attracteurs continus). Ces champs de neurones ont des propriétés de calcul variées mais sont particulièrement adaptés aux représentations spatiales et aux fonctions des étages précoces du cortex visuel. Ils ont été utilisés entre autres dans des applications de robotique autonome, dans des tâches de classification et clusterisation.

Comme de nombreux modèles de calcul neuronal, ils sont également intéressants du point de vue des architectures matérielles en raison de leur robustesse au bruit et aux fautes. On voit donc l'intérêt que ces modèles de calcul peuvent avoir comme solution permettant de dépasser (ou poursuivre) la loi de Moore. La réduction de la taille des transistors provoque en effet beaucoup de bruit, de même que la relaxation de la contrainte de $\sim 0\%$ de fautes lors de la production ou du fonctionnement des circuits permettrait d'énormes économies.

Par ailleurs, l'évolution actuelle vers des circuits many-core de plus en plus distribués implique des difficultés liées au mode de calcul encore centralisés de la plupart des modèles algorithmiques parallèles, ainsi qu'au goulot d'étranglement des communications. L'approche cellulaire est une réponse naturelle à ces enjeux.

Partant de ces différents constats, l'objectif de cette thèse est de rendre possible les calculs et applications riches des champs neuronaux dynamiques sur des substrats matériels grâce à des modèles neuro-cellulaires assurant une véritable localité, décentralisation et mise à l'échelle des calculs. Cette thèse est donc une proposition argumentée pour dépasser les limites des architectures de type von Neumann en utilisant des principes de calcul neuronal et cellulaire. Nous restons cependant dans le cadre numérique en explorant les performances des architectures proposées sur FPGA. L'utilisation de circuits analogiques (VLSI) serait tous aussi intéressante mais n'est pas étudiée ici.

Les principales contributions sont les suivantes :

1) Calcul DNF dans un environnement neuromorphique ; 2) Calcul DNF avec communication purement locale : modèle RSDNF (randomly spiking DNF) ; 3) Calcul DNF avec communication purement locale et asynchrone : modèle CASAS-DNF (cellular array of stochastic asynchronous spiking DNF).

Mots-clés: Champs neuronaux dynamiques, calcul cellulaire, FPGA, ingénierie neuromorphique, calcul stochastique.

Abstract

In the constant search for design going beyond the limits of the von Neumann architecture, non conventional computing offers various solutions like neuromorphic engineering and cellular computing. Like von Neumann who roughly reproduced brain structures to design computers architecture, neuromorphic engineering takes its inspiration directly from neurons and synapses using analog substratum. Cellular computing influence comes from natural substratum (chemistry, physic or biology) imposing locality of interactions from which organisation and computation emerge.

Research on neural mechanisms was able to demonstrate several emergent properties of the neurons and synapses. One of them is the attractor dynamics described in different frameworks by Amari with the dynamic neural fields (DNF) and Amit and Zhang with the continuous attractor neural networks.

These neural fields have various computing properties and are particularly relevant for spatial representations and early stages of visual cortex processing. They were used, for instance, in autonomous robotics, classification and clusterization.

Similarly to many neuronal computing models, they are robust to noise and faults and thus are good candidates for noisy hardware computation models which would enable to keep up or surpass the Moore law. Indeed, transistor area reductions is leading to more and more noise and the relaxation of the approx. 0% fault during production and operation of integrated circuits would lead to tremendous savings.

Furthermore, progress towards many-cores circuits with more and more cores leads to difficulties due to the centralised computation mode of usual parallel algorithms and their communication bottleneck. Cellular computing is the natural answer to these problems.

Based on these different arguments, the goal of this thesis is to enable rich computations and applications of dynamic neural fields on hardware substratum with neuro-cellular models enabling a true locality, decentralization and scalability of the computations. This work is an attempt to go beyond von Neumann architectures by using cellular and neuronal computing principles. However, we will stay in the digital framework by exploring performances of proposed architectures on FPGA. Analog hardware like VLSI would also be very interesting but is not studied here.

The main contributions of this work are : 1) Neuromorphic DNF computation ; 2) Local DNF computations with randomly spiking dynamic neural fields (RSDNF model) ; 3) Local and asynchronous DNF computations with cellular arrays of stochastic asynchronous spiking DNFs (CASAS-DNF model).

Keywords: Dynamic neural fields, cellular computing, FPGA, neuromorphic engineering, stochastic computing.