Université catholique de Louvain

Ecole Polytechnique de Louvain

LELEC 2990 - Travail de fin d'études

Conception d'une Plateforme SoC de Traitement de Signaux Numériques à Ultra-Basse Consommation pour la Détection des Crises d'Epilepsie

Jury :
Prof. David Bol (Promoteur)
Dr. Angelo Kuti
Prof. Jean-Didier Legat

Mémoire présenté en vue
de l'obtention du grade
d'ingénieur civil électricien
par Ludovic Moreau

Année académique 2013-2014

Table des matières

Liste des abréviations

- **AL** : *Active Learning*
- **ARM (Ltd)** : *Advanced RISC Machine*
- **ASIC** : *Application-Specific Integrated Circuit*
- **AFE** : *Analog Front-End*
- **BMI** : *Brain-Machine Interface*
- **BSN** : *Body Sensor Node*
- **CAD** : *Computer-Aid Design*
- **CHB** : *Children's Hospital Boston*
- **CMOS** : *Complementary Metal-Oxyde Semiconductor*
- **DCU** : *Data-Computation Unit*
- **DSP** : *Digital Signal Processor*
- **ECG** : *Electrocardiogram*
- **(s)/(i)EEG** : *(scalp)/(intracranial) Electroencephalogram*
- **FFT** : *Fast Fourier Transform*
- **FIR** : *Finite Impulse Response*
- **FN** : *False Negative*
- **FV** : *Feature Vector*
- **FP** : *False Positive*
- **FPGA** : *Field-Programmable Gate Array*
- **FSM** : *Finite-State Machine*
- **GPP** : *General-Purpose Processor*
- **HW** : *Hardware*
- **IC** : *Integrated Circuits*
- **IoT** : *Internet-of-Things*
- **Lat** : *Latence (d'un détecteur)*
- **L/S/HVT** : *Low/Standard/High Treshold Voltage*
- **LP** : *Low Power*
- **LNA** : *Low-Noise Amplifier*
- **MAC** : *Multiply-and-Accumulate*
- **MIT** : *Massachussets Institue of Technology*
- **MCU** : *Microcontroller Unit*
- **NVIC** : *Nested Vectored Interrupt Controller*
- **Pr** : *Précision (d'un détecteur)*
- **RBF** : *Radial-Basis Function*
- **RISC** : *Reduced Instruction Set Computing*

- **SAR** : *Successive Approximation Register*
- **Se(D)/(E)** : *Sensibilité (pour les détecteurs de Déclenchements/d'Evénements)*
- **SoC** : *System-on-Chip*
- **Sp(D)/(E)** : *Specificité (pour les détecteurs de Déclenchements/d'Evénements)*
- **SRAM** : *Static Random-Access Memory*
- **SVM(A)** : *Support Vector Machine (Accelerator)*
- **SW** : *Software*
- **TN** : *True Negative*
- **TP** : *True Positive*
- **ULP** : *Ultra Low Power*
- **WIC** : *Wakeup Interrupt Controller*
- **WSN** : *Wireless Sensor Node*

Introduction

Le déploiement global de l'informatique et des technologies de communication associées a impacté nos sociétés modernes à tous les niveaux. Du point de vue du grand public, il est indéniable que l'informatique - via l'Internet - est une révolution majeure, par son omniprésence quotidienne comme interface avec le monde : pour travailler ou se divertir, s'informer ou s'exprimer, voter ou faire campagne, acheter ou vendre des biens et des services, etc. L'utilisateur n'a généralement pas conscience que la réelle innovation technologique derrière cette évolution nous vient de l'électronique, et plus particulièrement du circuit intégré (IC), réalisé pour la première fois en 1958 [1]. A plus forte raison, il s'agit du nombre exponentiellement croissant de dispositifs semi-conducteurs intégrables sur une puce.

Cette croissance, gouvernée par la "célèbre" loi de Moore établie en 1965 [2], a permis au fil des années une augmentation importante des fonctions intégrables sur IC, à coûts réduits [3]. C'est la clé derrière l'évolution de l'informatique, caractérisée, elle, par la loi de Bell [4]. Cette loi décrit comment les différentes générations d'ordinateurs se sont succédées : depuis l'unité centrale pour une grosse entreprise dans les années 60, aux systèmes électroniques grand public que l'on connaît actuellement et représentant plusieurs unités par personne (smartphones, tablettes, etc.). Dans cette continuité, la prochaine génération devrait être constituée de plusieurs dizaines de milliards de capteurs sans-fil (WSN) millimétriques.

L'un des projets majeurs autour de l'avènement des WSN est le déploiement de l'Internet des objets (IoT), qui vise à développer une intelligence ambiante en attachant à chaque objet un noeud de capteur [5]. Toutefois, et à l'instar des précédentes classes d'ordinateurs, une telle perspective requiert des avancées technologiques majeures dans la conception des IC. En effet, la cible de développement réside dans des *systems-on-chip* (SoC) ayant certaines capacités de détection et de calcul, de la mémoire et des moyens de communication sans-fil, mais étant hautement contraints en termes de consommation, de taille, et de coût.

Dans cette vision des choses, le corps humain devrait également être équipé d'un réseau de capteurs (BSN). L'idée est d'exploiter les paramètres physiologiques du corps humain pour notamment réaliser du monitoring médical temps-réel, de la stimulation sensorielle (vue, ouïe), ou encore supporter le mouvement de membres prosthétiques [6]. Parmi d'autres applications possibles, l'amélioration de la santé et des soins de santé est une motivation évidente au développement des BSN. Or, dans ce type de systèmes, les interfaces avec le cerveau seraient prépondérantes.

Les interfaces cerveau-machines (BMI) – ou interfaces neuronales – représentent un espoir d'avancée technologique majeur du 21ème siècle, celui de connecter et interpréter les états intentionnels du cerveau [7]. En effet, depuis quelques années, les applications commerciales et publications académiques sur le sujet se multiplient dans plusieurs domaines. Etant donné l'interface avec le corps humain, le

domaine biomédical est évidemment au cœur du sujet : que ce soit dans un cadre prosthétique, thérapeutique ou simplement de diagnostic et surveillance.

Dans ce contexte, la détection des crises d'épilepsie via l'analyse des signaux électriques du cerveau (EEG) effectuée par des systèmes électroniques embarqués a reçu récemment une attention particulière [27–33]. En effet, l'épilepsie est le troisième trouble neurologique le plus répandu au monde, et un patient sur trois reste sujet à des attaques fréquentes - malgré un traitement médical - limitant son indépendance et sa mobilité et pouvant se traduire par une isolation sociale et/ou plus directement à des dommages physiques [8]. Cette raison a motivé le développement de systèmes thérapeutiques portables, capables de détecter et de réagir au déclenchement d'une crise d'épilepsie et, le cas échéant, d'administrer une thérapie locale par le biais d'un stimulus électrique, thermique ou neurochimique [9].

Le rôle de l'électronique est donc majeur dans ce type de systèmes embarqués. Typiquement, les signaux électriques neuronaux sont enregistrés via des électrodes, puis traités par un module analogique de conditionnement (amplification, filtrage et numérisation) avant d'être analysés par un dispositif digital [10] (voir Fig. 1). Ce dernier exécute des opérations classiques de traitement de signaux dans le but de les discriminer comme étant normaux ou révélant une crise imminente ou en cours. Plus particulièrement, le signal courant est caractérisé par le biais de "*features*" (spectraux, spatiaux, temporels) ayant une corrélation forte avec l'état physiologique cible et formant un vecteur de données soumis à un classificateur binaire supervisé [11].

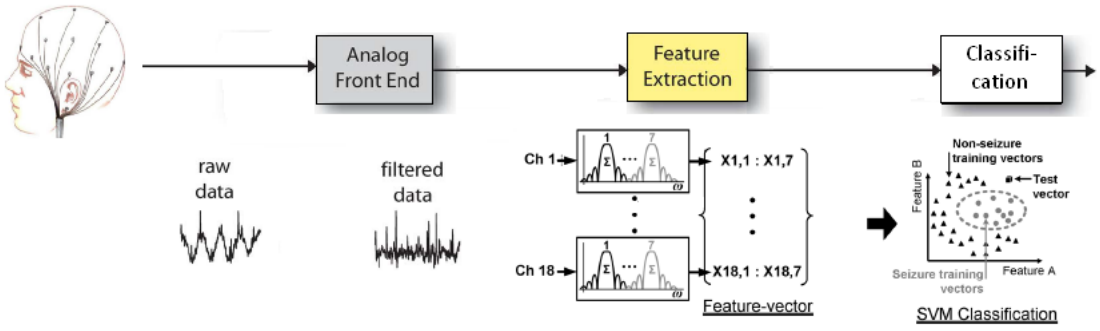


Figure 1 – Chaîne de traitement typique pour la détection du déclenchement des crises d'épilepsie à partir de signaux électroencéphalogrammes ([10], [27])

Dans ce type de systèmes, la transmission sans-fil – nécessaire – des données représente un coût important en termes de consommation [27]. Or celle-ci est contrainte d'une part par l'alimentation sur batterie et d'autre part l'exposition des tissus cellulaires à la chaleur dissipée [7] (1-10 mW pour les systèmes externes [21]). Depuis quelques années, deux optiques se sont développées pour réduire la consommation générale en diminuant le taux de transmission de données. La première se concentre sur la compression de données après extraction des "*features*" [12]. D'autre part, les avancées des algorithmes de calcul au niveau hardware (HW) ont permis d'intégrer au système embarqué l'opération de classification. Celle-ci est généralement issue d'une technique de "*machine learning*". Or, dans le cadre de la détection du déclenchement des crises d'épilepsie, ce sont les machines à vecteurs de supports (SVM) qui prédominent [27, 29, 30, 33]. Enfin, il est à noter que certains travaux sur le "*compressive*

sensing" combine les deux approches [28].

L'objectif de cette recherche est la conception et l'optimisation de circuits intégrés digitaux de type *system-on-chip* (SoC) destinés à réaliser la partie de traitement de signal numérique d'un système électronique embarqué de détection du déclenchement de crises d'épilepsie à partir de signaux électroencéphalogrammes. De façon à satisfaire aux contraintes énergétiques mentionnées précédemment, l'accent sera placé sur la minimisation de la consommation électrique. Le challenge réside donc dans l'accomplissement de cet objectif de basse consommation sans nuire aux performances du détecteur (latence, sensibilité et spécificité). Dans cette optique, une attention particulière sera portée au module de classification, actuellement le plus coûteux en terme de consommation.

Les contributions principales de ce travail sont :

- La reformulation des étapes critiques (d'un point de vue calculatoire) d'un algorithme de détection du déclenchement des crises d'épilepsie déjà éprouvé par la littérature (développé par Shoeb [11]), pour le portage sur un SoC à faible consommation en minimisant l'impact sur les performances du détecteur.

- La validation de cet algorithme pour plusieurs patients de la base de données CHB-MIT selon un schéma de validation croisée "*leave-one-out*", et la mise en avant des meilleures performances données par les noyaux SVM de complexité inférieure (linéaire, polynomial de degré 2).

- La validation à titre indicatif de l'utilisation d'un algorithme de sélection de *features* de type "*greedy search*" pour déterminer un sous-set pertinent de canaux sEEG spécifique au patient.

- Le développement d'une plate-forme SoC autour d'un microprocesseur ARM Cortex-M0 tirant profit des *features* de basse-consommation de ce dernier.

- Le portage *software* du détecteur complet sur ce SoC et la validation des optimisations algorithmiques réalisées, rendant possible la réduction de la fréquence d'opération du système sous la contrainte temps-réel de traitement successif des segments sEEG de 2 secondes.

- La conception d'un accélérateur SVM modulable pour les noyaux linéaire et polynomial de degré 2, puisque validés comme donnant de manière générale les meilleures performances de détection pour l'algorithme de Shoeb. L'accent a été placé sur la concision du module et donc sur la minimisation de sa consommation.

- La synthèse pour une technologie digitale CMOS bulk 65nm LPSVT (STMicroelectronics) du SoC et la caractérisation de la consommation pour le détecteur à noyau linéaire.

Enfin, la suite du texte est organisée comme suit : dans un premier chapitre, nous partirons du contexte de l'épilepsie pour poser les motivations au développement d'un SoC à basse-consommation pour la détection du déclenchement des crises d'épilepsies. En outre, nous y justifierons le cadre de travail sur base des différentes techniques de détection possibles et de l'état de l'art, lequel nous permettra de mettre en évidence un algorithme particulièrement pertinent.

Ensuite, nous détaillerons cet algorithme dans un second chapitre. Nous verrons comment les deux phases principales (extractions des *features* et classification) sont réalisées, et nous discuterons les différentes paramètres liées à celles-ci.

Un troisième chapitre nous permettra de présenter une première implémentation *software* de cet algorithme via Matlab. D'emblée, le développement sera réalisé dans l'optique du portage subséquent sur la plate-forme SoC cible. Plusieurs optimisations dans ce sens seront portées et validées pour différents patients de la base données CHB-MIT.

Enfin, nous détaillerons le SoC développé dans un quatrième et dernier chapitre. Après une vue d'ensemble du système, nous présenterons le processeur Cortex-M0, en mettant en évidence sa pertinence pour son intégration à des plate-formes telles que la nôtre, ainsi que la façon dont nous avons pu exploiter ses *features* de basse-consommation. Ensuite, nous verrons comment le portage du détecteur logiciel a pu être réalisé, et nous validerons les différentes optimisations portées. Et pour terminer, nous présenterons la conception de l'accélérateur SVM, ainsi que les résultats de synthèse.

Chapitre 1

Contexte

L'objectif de ce chapitre est de fournir les notions fondamentales nécessaires à la compréhension d'une part, des motivations de ce travail et, d'autre part, de l'algorithme utilisé et décrit au chapitre 2. Dans cette optique, nous donnerons ici dans un premier temps, une vision brève mais précise de ce qu'est l'épilepsie. Ensuite, nous discuterons les différentes façons de réaliser une détection d'attaque épileptique et nous définirons, à cet égard, le cadre du présent travail. A partir de ce cadre, nous reprendrons alors les différents éléments typiques de la chaîne de traitement des signaux électriques du cerveau et spécifie ceux visés par ce travail. Enfin, l'état de l'art du domaine sera présenté, suivi d'un résumé succinct de cette partie qui nous permettra de repréciser les objectifs de ce travail.

1.1 Épilepsie

A l'instar de nombreux troubles de santé, l'épilepsie a été considérée pendant des siècles comme ayant des origines occultes. Son étymologie vient du Grec, "*epilēpsia*", pouvant se traduire par "se saisir de" [14]. En d'autres termes, on pensait que la personne affectée était saisie par une force supérieure.

L'épilepsie est maintenant définie comme un trouble neurologique caractérisé par une tendance accrue à être sujet à des attaques récurrentes [15]. Il ne s'agit pas d'une maladie en tant que telle ; en réalité, les attaques sont des anomalies cérébrales causées par des maladies ou des traumatismes. En pratique, l'étiologie peut être connue (symptomatique), mais reste inconnue dans près de 70% des cas (idiopathique), accordant encore une once de crédit aux Grecs Anciens. Dans le cas d'une épilepsie symptomatique, la cause peut varier de manière significative avec l'âge et la physiologie, restant toutefois de l'ordre d'un dérangement cérébral (p.ex. neurodégénérescence génétique, maladie cardiovasculaire, trauma crânien, etc.).

Les attaques épileptiques peuvent être définies comme *une occurrence transitoire de signes et/ou de symptômes dus à une activité neuronale anormale excessive et synchrone* [16]. Son aspect transitoire, avec un début et une fin "marquées", implique quatre états pour le patient : preictal, ictal, postical (respectivement juste avant, pendant et un peu après (10-30 min.) une attaque [1]) et interictal (entre les attaques). En outre, le côté anormalement synchrone de l'activité cérébrale permet de détecter les périodes ictales sur base de signaux électroencéphalogrammes (EEG).

Il existe une large variété de manifestations possibles pour une attaque épileptique. Cela peut affecter un ou plusieurs facteurs parmi les fonctions sensorielles ou motrices, l'état de conscience ou émotionnel, la mémoire ou le comportement. Cependant, les convulsions sont le symptôme épileptique le plus connu et le plus courant. Et, même s'il existe des déclencheurs notables tels que l'alcool, le

1. NB : dans la suite du texte, le terme attaque se référera toujours à une attaque épileptique.

déficit de sommeil ou le stress, les attaques sont généralement soudaines et aléatoires.

Cette condition de santé affecte près de 1% de la population mondiale, ce qui en fait le troisième trouble neurologique le plus répandu après la maladie d'Alzheimer et les AVC's [8]. Et dans près de 30% des cas, les attaques persistent malgré un traitement médical ou une chirurgie : on parle alors d'attaques "médicalement intraçables" ou encore d'épilepsie "réfractaire". Or, il est évident que ce type d'attaques limite fortement l'autonomie et la mobilité de l'individu : une large gamme d'activités à priori anodines telles que le simple fait de conduire ou prendre un bain deviennent sujettes à un danger permanent. Et même en se montrant précautionneux, les attaques représentent toujours un risque de dommages physiques dans des situations de la vie quotidienne (chute, brûlure, coupure, etc.), allant, dans le cas extrême, jusqu'à une mort soudaine de la personne atteinte [2] [9].

L'épée de Damoclès que cela représente met en lumière le besoin pour de meilleurs moyens de contrôle et de diminution des attaques. Depuis une quinzaine d'années, de nombreux efforts ont été réalisés dans le développement de nouveaux systèmes thérapeutiques capables de détecter et de réagir au déclenchement d'une attaque, en administrant - le cas échéant - un stimulus local (électrique, thermique ou neurochimique) de façon à stopper l'attaque avant que n'apparaissent les symptômes cliniques. Ce type de systèmes ouvre le champ à bien des perspectives : par exemple, on peut imaginer un système d'alerte qui permettrait au patient de se mettre dans une position sécurisée ou encore qui préviendrait des proches. Enfin, il est à noter que ces systèmes de détection se révèlent pertinents également pour les cas non-réfractaires nécessitant une prise quotidienne de médicaments anti-épileptiques : afin de réduire d'une part la charge que cela représente, mais aussi et surtout les effets secondaires pouvant se révéler toxiques.

1.2 Détection d'attaques par électroencéphalogramme

Un cerveau humain peut être vu comme un ensemble de réseaux neuronaux inter-connectés. Ces réseaux sont composés de neurones capables de générer, propager et traiter des signaux électriques [8]. Or les attaques épileptiques sont marquées par une activité et un synchronisme accrus d'un large groupe de neurones parmi un ou plusieurs de ces réseaux. Il est donc possible de détecter une attaque en mesurant l'activité électrique du cerveau.

Le moyen de mesure le plus répandu est l'électroencéphalogramme, soit en répartissant plusieurs électrodes de manière symétrique à la surface du cerveau (scalp/surface - sEEG), soit en implantant une ou plusieurs électrodes à l'intérieur du cerveau (intracrânien/invasif - iEEG [3]). En outre, les détecteurs d'attaque peuvent être différenciés par leur objectif : les "détecteurs de déclenchement" dont le but est d'identifier le début d'une crise épileptique avec le plus court délai possible mais en relâchant la contrainte de précision, et les "détecteurs d'événement" dont le but est d'identifier de façon

2. 1 décès par 200 cas par an dans les cas d'épilepsie réfractaire
3. NB : dénommé également électrocorticogramme (ECoG)

aussi précise que possible le début et/ou la fin des attaques, mais sans contrainte spécifique sur le délai.

Que ce soit pour le type d'EEG utilisé ou de détecteur implémenté, nous verrons dans quel cas chaque solution est appropriée, et sur lesquelles nous travaillerons ici.

1.2.1 Types d'attaques épileptiques

Les deux types principaux d'attaques épileptiques sont dites "partielles" (focales) ou "généralisées", selon qu'elles se concentrent sur une partie du cerveau ou sur son ensemble [17]. A noter que dans certains cas, une attaque peut démarrer de manière locale puis se répandre et est alors qualifiée de "secondairement généralisée".

Dans le **cas partiel**, les symptômes cliniques sont logiquement liés à la zone du cerveau concernée. Par exemple, si l'attaque se produit dans le lobe temporel - régissant les émotions et la mémoire à court-terme - elle peut, par exemple, provoquer de manière inopinée un sentiment d'euphorie ou de peur, des déjà-vus ou encore des hallucinations de goût, d'odeur [8].

Par ailleurs, dans le **cas généralisé**, les attaques peuvent se classer de différentes façons selon qu'il y ait convulsion du corps entier ou non, et selon l'état de conscience de l'individu pendant son déroulement. A noter qu'il existe également un cas particulier où il n'y a ni convulsion, ni altération de conscience, mais plutôt une perte soudaine du tonus musculaire sur le corps entier menant irrémédiablement à la chute.

1.2.2 Types de détecteurs

Dans le cas des **détecteurs de déclenchement**, l'intérêt de l'analyse des signaux électriques se trouve dans le fait qu'il existe généralement un certain laps de temps entre le déclenchement électrique [4] d'une crise (notable via les signaux mesurés), et son déclenchement dit clinique (symptômes manifestes) - voir Fig. 1.1. Les applications peuvent être de trois ordres : diagnostic (p.ex. activation au moment opportun d'un système de neuro-observation capable de tracer la source profonde d'une crise), thérapie (p.ex. activation d'un stimulus capable de réduire ou d'annihiler l'attaque) et alerte (p.ex. de façon à permettre à l'individu de se mettre à l'abri ou de s'administrer un anti-convulsant rapide). Dans les trois cas, la minimisation du délai entre le déclenchement électrique et la détection est primordiale, mais nuancée par le niveau de précision temporelle (sur le début de la crise) requis par l'application. En d'autres termes - et de manière intuitive - détecter le début de la crise avec une précision importante requiert plus de calculs et donc nécessite de relâcher la contrainte de latence.

D'un autre côté, les **détecteurs d'événements** épileptiques requièrent l'identification la plus

4. NB : dans la suite du texte et à moins qu'il ne soit précisé autrement, le terme déclenchement fera référence au déclenchement électrique (ou électrographique) d'une attaque épilpetique

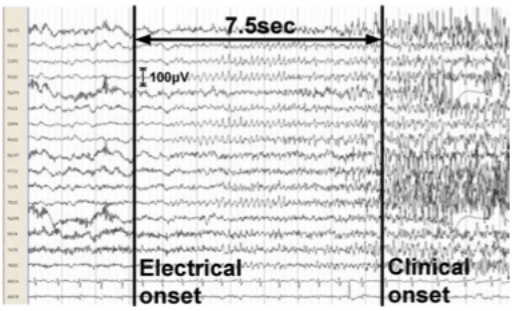


Figure 1.1 – sEEG 18 canaux montrant le délai entre le déclenchement électrique d'une crise épileptique et son déclenchement clinique (marqué par des réflexes musculaires, causant les larges variations dans les signaux) [27]

précise possible du début et (généralement) de la fin d'une attaque. L'attrait principal à cela réside dans la possibilité pour le médecin d'ajuster au mieux le traitement dans le cas des épilepsies non-réfractaires. En effet, celui-ci est généralement établi sur base du nombre et de la sévérité des attaques subies et rapportées - mais souvent de manière imprécise - par le patient. Or, un traitement sur-dosé ou sous-dosé peut mener respectivement à une augmentation des effets secondaires ou simplement à une persistance de la fréquence des attaques.

Les deux types de détecteurs présentent un intérêt. Toutefois étant donné les contraintes et les objectifs différents, les algorithmes utilisés - bien que reposant sur les mêmes bases théoriques - ne sont généralement pas les mêmes. Dans le cas présent, nous nous intéresserons à la détection du déclenchement des crises d'épilepsie.

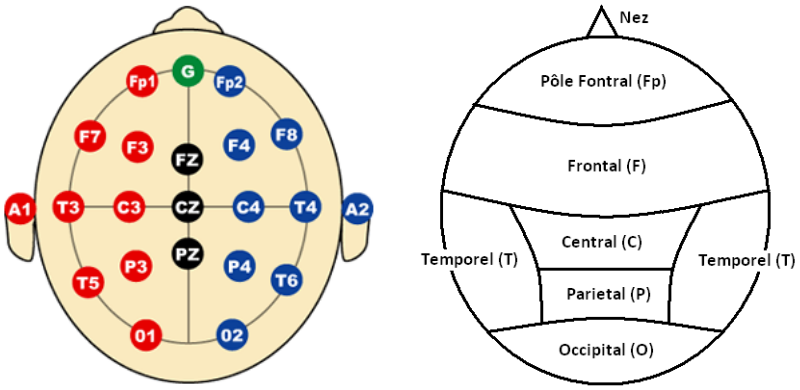
1.2.3 Electroencéphalogramme (sEEG/iEEG)

Une électrode mesure le potentiel électrique généré par plusieurs millions de neurones [11]. Un signal - ou canal - EEG est réalisé via la différence de potentiel entre deux électrodes.

La Figure 1.2a reprend une disposition typique des électrodes pour le **sEEG** : chacune résume l'activité d'une région du cerveau (voir Fig. 1.2b). Toutefois, les signaux électriques mesurés de cette façon sont contraints à deux égards : d'une part, les neurones les plus proches du scalp y contribuent plus que les autres tandis que ceux enterrés profondément ne sont parfois pas observables. D'autre part, le fluide cérébrospinal et le squelette entourant le cerveau atténuent grandement l'amplitude du signal pour les "hautes" fréquences (30 à 100Hz).

A l'inverse, un implant **iEEG** permet de mesurer l'activité d'une population réduite de neurones (augmentant la résolution spatiale) et pouvant être plus profondément ancrée dans le cerveau. En outre, le signal n'est alors corrompu ni par les artefacts physiologiques mentionnés plus haut, ni par d'autres interférences dont souffre le sEEG (liées à "l'environnement").

Il est évident que les avantages apportés par l'iEEG sont contrebalancés par le fait d'être restreint



(a) (b)

FIGURE 1.2 – (a) Représentation du système international "10-20" de placement des électrodes pour un sEEG (Source : http ://www.immrama.org/eeg/electrode.html). (b) Représentation des régions du cerveau.

à une zone précise du cerveau, limitant la détection des attaques partielles : celles qui trouveront leur origine dans une autre zone ne seront pas détectable. De plus, l'iEEG nécessite une intervention chirurgicale limitant les zones d'implant possibles.

Par ailleurs, l'iEEG permet la détection du déclenchement électrique d'une attaque parfois plusieurs dizaines de secondes avant le sEEG. Toutefois, cette amélioration est contrastée par le fait que l'iEEG identifie également d'autres phénomènes neurologiques qualifiables d'anormaux mais indépendants des crises d'épilepsies et pouvant mener à une confusion dans la détection.

A l'instar des types de détecteurs, les deux façons de réaliser l'EEG présentent un intérêt, selon l'application recherchée. Toutefois cela modifie les contraintes du système. En plus des différences notables dans la mesure réalisée mentionnées aux paragraphes précédents, la consommation n'est pas restreinte de la même manière : d'une part en raison du fait qu'un implant doit également occuper moins de surface, ce qui limite la taille de batterie possible, mais surtout en raison de l'exposition plus ou moins importante des tissus cérébraux (limite à $800\mu\text{W}/mm^2$ pour les implants [10]). Typiquement, le budget en puissance pour les systèmes externes est de 1-10 mW, et de 10-100 μW pour les systèmes internes [21]. Par conséquent, un SoC implantable n'aura pas la même capacité de calcul disponible, mais doit traiter à priori moins de données et de meilleure "qualité" (ce qui lui permet d'utiliser des modèles d'ordre moins important) [11].

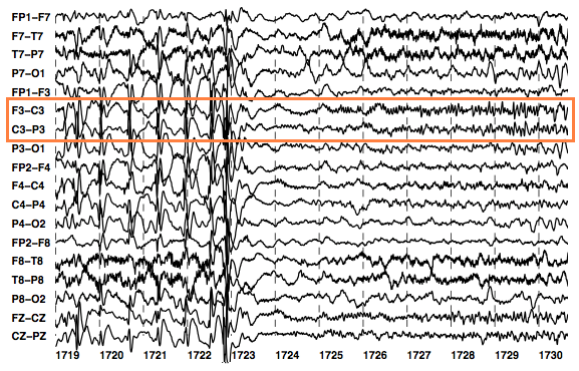
Dans le cas présent, nous utiliserons des données sEEG en raison de la disponibilité d'une large base de données gratuite, fiable et déjà éprouvée par plusieurs recherches en traitement de signaux, ce qui en fait un cadre idéal pour le portage d'algorithmes *software* (SW) dans un système *hardware* (HW) optimisé. Il s'agit de la base de données "CHB-MIT" introduite dans [8] et disponible via [18,19] : elle sera décrite à la partie 3.1.

1.2.4 Détection des attaques

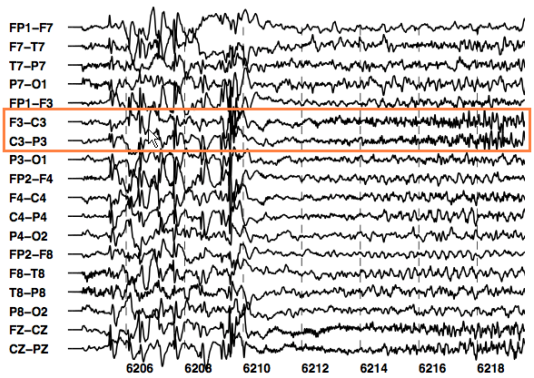
Dans le cas sEEG, la gamme de fréquences pertinente pour la détection d'une activité épileptique est la bande 0.5-25Hz [17]. Selon la composante fréquentielle f dominante, l'on parlera d'onde "delta" ($f \leq 4$Hz), "theta" ($4 < f < 8$Hz), "alpha" ($8 \leq f < 12$Hz), "beta" ($12 \leq f < 30$Hz) ou gamma ($f > 30$Hz) [9]. L'activité cérébrale mesurée par sEEG est également caractérisée selon sa distribution spatiale.

Comme mentionné précédemment, la détection des attaques épileptiques par analyse des signaux électriques du cerveau est rendue possible par une soudaine redistribution de l'énergie spectrale sur plusieurs canaux, causée par l'hyper-synchronisme des neurones touchés. En d'autres termes, une activité rythmique similaire se développe sur plusieurs canaux EEG, correspondant à l'apparition ou à la disparition de composantes fréquentielles dans la bande d'intérêt [11]. Il est important de noter que les canaux EEG concernés ainsi que les fréquences prédominantes lors des crises épileptiques sont très variables d'un patient à l'autre. Cependant, ces caractéristiques affichent une corrélation importante pour un même patient lorsque l'attaque émerge de la même zone du cerveau, ce qui motive le développement de détecteurs spécifiques [5] à chaque patient.

La Figure 1.3 montre les signaux sEEG pour un même patient lors de deux attaques différentes et permet d'observer d'une part l'activité rythmique qui se développe (particulièrement aux canaux F3-C3 et C3-P3) et d'autre part la similarité tant d'un point de vue spatial et spectral que temporel.



(a) Attaque à partir de 1723s [11]



(b) Attaque à partir de 6210s [11]

Figure 1.3 – Extrait des signaux sEEG d'un même patient lors de deux attaques différentes

1.3 Système de détection

A présent que le cadre de travail est précisé, il est intéressant de se pencher plus en avant sur les aspects "système" du problème qui nous occupe : la détection du déclenchement des crises d'épilepsie par le biais de signaux sEEG. Comme entrevu précédemment, l'objectif ultime est le développement

5. NB : dans la suite de ce texte, la dénomination "détecteur spécifique" référera au fait qu'il est spécifique à chaque patient.

d'un système électronique complet et autonome. Nous verrons dans cette partie quels sont les challenges que cela incombe et, à partir de là, quelle est l'architecture haut-niveau typique d'un système performant. Enfin, il sera précisé sur quelle partie se concentre ce travail.

1.3.1 Challenges associés

Tout d'abord, un problème inhérent à l'épilepsie (et donc indépendant du type d'EEG) et décrit à la partie précédente est la variabilité des attaques entre les personnes atteintes, au niveau de l'activité cérébrale (d'un point de vue spatial et fréquentiel). Cela oblige à traiter le problème de manière spécifique et empêche le développement de solutions entièrement génériques. En outre, il est à noter que l'activité électrique du cerveau est modulée par l'état de vigilance de l'individu (p.ex. éveillé ou endormi) : là aussi, la composante fréquentielle dominante et la distribution spatiale (i.e. entre les différents canaux) de l'énergie spectrale en sera modifiée [17]. En d'autres termes, le système doit être capable de discriminer d'autres types d'activité rythmique qui ne sont pas le reflet d'une attaque sous-jacente.

Ensuite, nous avons vu que les signaux sEEG permettent une couverture complète de la surface du cerveau mais, étant mesurés de manière externe, ils sont faibles (quelques μV) et sensibles à bon nombre d'interférences physiologiques (p.ex. transpiration, clignement d'yeux, contractions musculaires, etc.) et non-physiologiques (p.ex. bruit d'alimentation, interface peau-électrode, mouvement des câbles, etc.) [8].

Par ailleurs, l'EEG d'un patient épileptique varie constamment d'un état de régime à un état transitoire, rendant le processus non-stationnaire et forçant une caractérisation à court-terme de l'évolution de l'activité cérébrale. La Figure 1.3 permet d'observer cela : à partir de l'attaque (1723s) et sur l'espace de quelques secondes, le signal croît en amplitude et décroît en fréquence de façon significative. De plus, étant donné que les attaques sont des événements "rares" [6], les données caractérisant les "états de crise" le sont également par rapport aux données en "état normal".

Enfin, afin de développer un système électronique portable viable, il est nécessaire de le réaliser sous forme de circuit le plus intégré possible, hautement contraint en termes de complexité, de taille et de consommation (typiquement 1-10mW [21]).

1.3.2 Un modèle basé sur les données

En raison des problèmes sus-cités du sEEG (mesure externe, interférences) et de la variabilité entre les patients, il s'avère généralement très limitant - d'un point de vue précision - d'utiliser des modèles basés sur la physiologie [21], tentant de caractériser de manière analytique des processus biologiques

6. C'est-à-dire même lorsqu'un patient est sujet à des crises fréquentes, cela reste rare d'un point temporel par rapport aux états normaux

complexes. Par conséquent, les solutions basées sur la modélisation à partir de données ont le vent en poupe depuis plusieurs années : d'une part en raison de la meilleure capacité des hôpitaux à générer, expertiser et stocker des données pour chaque patient traité, et d'autre part, en raison de l'évolution des techniques de "*machine learning*" dans l'analyse des données. En effet, l'efficacité et la rapidité de telles techniques dans le développement de modèle quantitatifs sur base de larges bases de données permet, avec un même système, de développer des solutions spécifiques pour chaque patient pour autant que des données pertinentes associées soient disponibles.

Dès lors, la plupart des algorithmes récents de détection pour des applications biomédicales reposent sur ce principe. Ils requièrent deux aspects principaux : l'extraction de bio-marqueur, et leur interprétation. Ces bio-marqueurs sont des paramètres des signaux pour lesquels il existe une corrélation forte avec l'état physiologique cible du patient [21]. Par exemple, dans le cas qui nous intéresse (crises d'épilepsie) - et comme déjà mentionné plus haut - il existe des similarités notoires dans la distribution spatiale et temporelle de l'énergie spectrale entre différentes attaques pour un même patient. Et lorsque ces bio-marqueurs sont traités comme un vecteur de *features* (FV), l'interprétation peut être réalisée par un classificateur binaire de type "*machine-learning*". En d'autres termes, il s'agit d'une classification par apprentissage - ou supervisée - nécessitant au préalable une phase d'entraînement sur base de données représentant les différents états que l'on souhaite identifier. Enfin, il est intéressant de noter que des classificateurs tels que les machines à vecteurs de support (SVM) sont particulièrement pertinents pour le problème qui nous occupe : en effet, ils donnent de bonnes performances lorsque d'une part les bases de données à traiter sont importantes, et d'autre part lorsqu'il y a une différence importante entre la quantité de données représentant chaque état à caractériser.

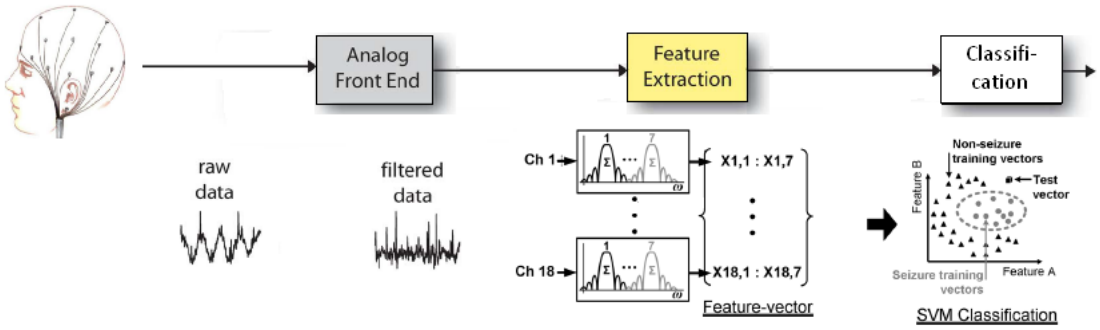


Figure 1.4 – Architecture haut-niveau d'une chaine de traitement typique des signaux sEEG pour la détection du déclenchement des crises d'épilepsie (éléments provenant de [10] et [27])

1.3.3 Chaine de traitement des signaux

Par rapport aux challenges établis à la partie 1.3.1, l'utilisation d'un modèle basée sur les données décrit à la partie 1.3.2 se justifie. En effet, celui permet de pallier notamment à la variabilité entre la patients, à la discrimination des autres activités induisant une rythmique notable dans les signaux

sEEG, à l'utilisation de set de données fortement déséquilibrés, etc. Toutefois, la mise en place d'un système complet nécessite de résoudre les autres problèmes également (faible intensité de signal, interférences, etc.).

La Figure 1.4 reprend l'architecture de haut-niveau typique d'une chaîne de traitement de signaux de détection du déclenchement des attaques épileptiques via un modèle basé sur les données. Les électrodes sont donc placées à la surface du crâne (sEEG). Les données brutes qu'elles enregistrent sont traitées par un premier étage de conditionnement (AFE - amplification, filtrage, numérisation). Ensuite, un module digital est chargée de réaliser un traitement numérique comprenant l'extraction des *features* et la classification : c'est sur cette partie uniquement que porte ce travail. Enfin, un système de communication sans-fil est généralement présent en pratique. Celui-ci est utilisé dans l'optique de réaliser un système en boucle fermée, par exemple pour transmettre un signal d'alerte en cas de détection (et activer un traitement en conséquence) ; une autre possibilité pour réaliser un système en boucle fermée consiste à récupérer les nouvelles données sEEG afin de faire évoluer le modèle [33].

Le module de traitement de signaux numériques peut être implémentée de trois manières [7]. Une première façon de procéder consiste à réaliser les opérations nécessaires entièrement en SW sur un processeur approprié (généralement un DSP), soit la solution notamment la plus facile à mettre en oeuvre mais au prix d'une consommation trop importante pour nos contraintes IC [38]. Une autre possibilité est d'utiliser uniquement des circuits HW dédicacés (partie 1.4.1), permettant d'optimiser au mieux la surface de silicium requise et la consommation, mais aux dépens de la flexibilité. Enfin, comme nous le verrons à la partie 1.4.2, certaines approches combinent les deux premières, offrant un compromis intéressant.

1.4 État de l'art

La détection des attaques épileptiques par le biais de signaux EEG a débuté avec les détecteurs dits d'événements, dans les années 70. Ces systèmes étaient alors non-spécifiques aux patients, et donc - comme expliqué plus haut - fortement limités en terme de précision. L'un des premiers exemples rapportés dans la littérature est celui de Gotman [22], en 1982. Peu de solutions ont été développées avant les années 90, où apparurent les premiers détecteurs spécifiques. En outre, le développement d'applications diagnostiques et thérapeutiques a motivé l'apparition de détecteurs de déclenchement des attaques, avec Qu et al. [23], dès 1993.

Depuis une quinzaine d'années, les algorithmes se sont multipliés de manière importante, variant les types de *features* et de classificateurs. Toutefois, tous ne sont pas pertinents pour un portage sur dispositif intégré alimenté sur batterie et donc hautement contraint en termes de taille et de consommation. En effet, nous verrons ci-dessous que peu de solutions IC existent à ce jour pour le problème

7. Sur un ASIC du moins.

qui nous occupe et que la majorité implémente le même algorithme. Ce dernier sera également utilisé dans ce travail et présenté au chapitre 2.

Cette partie vise donc à reprendre l'état-de-l'art des solutions IC réalisant une détection du déclenchement des attaques épileptiques sur base de signaux sEEG. Nous verrons deux types de solutions : une première série, entièrement dédiée à ce problème et donc très performantes mais peu flexibles, et une seconde, que nous qualifierons de plate-formes SoC biomédicales, construites autour d'un processeur à usage général (GPP), et équipées de différents accélérateurs HW réalisant une ou plusieurs opérations d'extraction des *features* (FE) et/ou de classification.

1.4.1 Solutions dédicacées

Le Tableau 1.1 compare les différentes solutions présentées dans cette partie. Il est à noter que les critères de performance du détecteur (sensibilité, spécifité et latence) seront décrits à la partie 2.2.

En 2010, **Verma et al.** [27] proposent un SoC faible-puissance destiné à l'acquisition de signaux sEEG et, sur base de ceux-ci, à la détection du déclenchement d'attaques épileptiques. Le principe de ce système est de placer un exemplaire de la puce développée sur chaque électrode, afin, notamment, d'en rapprocher le plus possible la circuiterie d'acquisition et de rendre ainsi la détection plus robuste aux interférences. Les données sont alors transmises à une unité centrale (portable également mais hors scalp). Mettant en exergue le fait que la communication sans-fil - nécessaire pour éviter tout risque de strangulation avec des câbles - domine largement la consommation de ce type de puce, les auteurs choisissent de réaliser l'extraction des *features* (FE) de manière locale dans l'optique de réduire le taux de transmission de données, et donc la consommation associée. Le SoC contient donc un premier étage analogique de conditionnement (AFE - amplificateur faible-bruit (LNA) d'instrumentation), un ADC (SAR 12-bit) et un processeur digital dédicacé pour le FE. L'algorithme utilisé est celui de Shoeb [11] : le classificateur est donc une SVM (noyau gaussien - RBF), implémentée à part sur une unité centrale (via un MSP430). La communication est réalisée par une radio hors-puce - commune à tous les canaux auxquels elle est reliée par câble sur le scalp - dont la consommation est réduite par un facteur 40 grâce au FE local.

Shoaib et al. [28] intègrent, en 2012, le module de classification à leur puce, mais en délaissant toutefois l'AFE. En outre, ils réalisent toutes les opérations dans un domaine compressé ("*compressive sensing*"). La conjonction de ces deux éléments leur permet de réduire la consommation des différentes opérations ainsi que de la communication des données. En effet, ils évitent les coûts énergétiques liés à une éventuelle reconstruction (décompression) dans la chaîne de traitement, tout en réduisant le taux de transmission. La puce est donc composée de trois modules HW - réalisant respectivement la compression, le FE, et la classification - ainsi que d'une banque de SRAM. L'algorithme utilisé est à nouveau celui de Shoeb [11]. Par contre, différents noyaux possibles pour la SVM sont implémentés

(linéaire, polynomial, RBF). Les auteurs démontrent les différences énergétiques importantes (jusqu'à un facteur 50) liés au noyau utilisé, ainsi qu'au facteur de compression des données, qui affecte également de manière importante les performances du détecteur.

Yoo et al. [29] proposent en 2013 un SoC intégrant une AFE à 8 canaux, un ADC (SAR 10-bit), le module FE et la classification (SVM). Une fois de plus, l'algorithme implémenté est celui de Shoeb [11]. Etant donné que le module SVM peut être partagé mais que le FE doit être réalisé pour chaque canal, l'accent est ici placé sur l'optimisation de ce dernier en termes de consommation et de taille par rapport à une solution complètement parallèle (au prix de la latence). En outre, dans une optique de réduction agressive de la consommation, le noyau de la SVM est restreint au cas linéaire ce qui dégrade de façon importante la sensibilité du détecteur. La même année, **Altaf et al.** [30] proposent une version améliorée de ce SoC, avec un module FE encore optimisé au même titre que la SVM (noyau gaussien log-linéarisé). En conséquence, les performances du détecteur sont rehaussées tout en ayant une consommation moins importante.

		Unités	Verma [27] **JSSC10**	Shoaib [28] **CICC12**	Yoo [29] **JSSC13**	Altaf [30] **ISSCC13**
Type détecteur/EEG		-	Déclenchement/sEEG			
Algorithme implémenté		-	Shoeb [11]			
Base de données		-	/	CHB-MIT [19]		
Particularité		-	Etat d'éveil [a]	CPF [b] : ξ = 2-24x	-	-
Technologie (CMOS)		-	180nm	130nm	180nm	180nm
Etage de condi. analog.		-	Oui	Non	Oui	Oui
Nombre de canaux		-	1	/	1-8	1-8
Extraction des *features*		-	Oui	Oui	Oui	Oui
Classification		-	Non	Oui	Oui	Oui
Noyau SVM		-	/	Configurable	Linéaire	Log-Lin. RBF
Surface de puce		mm^2	6.25	1.5	25	25
Cadencement		kHz	N/C	1020-300	64-512	N/C
Tension d'alimentation		V	1	1.2-0.44	1.8(A)/1.0(D)	1.8(A)/1.0(D)
Energie/Classification		μJ/FV	/	0.056 (SVM Lin.) 11.1-17.2 (SVM Poly4) 16.9-26.8 (SVM RBF)	2.03	1.83
Performances du détecteur	**Sensibilité**	%	/	96-80	84.4	95.1
	Spécificité	FP/h	/	0.15-0.79	N/C	N/C
	Latence	s	< 2.5	4.7-17.8	< 2	< 2

a. Algorithme de détection démontré pour un seul canal à la fois. Donc, bien que conçu pour les attaques épileptiques, les tests ne sont réalisés que pour détecter l'état d'éveil (yeux ouverts/fermés).

b. Facteur de compression dans l'utilisation du "*Compressive Sensing*" ; variable, donc induit une gamme de valeurs pour tous les autres paramètres.

Tableau 1.1 – Comparaison des solutions IC entièrement dédiées à la détection du déclenchement des attaques.

1.4.2 Plate-formes biomédicales

Nous considérons donc comme "plate-formes biomédicales" les SoC construits autour d'un GPP dans une optique de flexibilité et de modularité. Le Tableau 1.2 compare les différentes solutions présentées ici. Il est à noter que les performances des détecteurs n'y sont pas reprises car peu détaillées dans les articles relatifs, contrairement à ceux des solutions entièrement dédiées.

Un premier exemple est donné par **Sridhara et al.** [31] avec une puce composée notamment d'un GPP RISC 32-bit de type ARM Cortex-M3 et d'un accélérateur FFT. L'un des arguments des auteurs sur la viabilité de tels systèmes est que les signaux biomédicaux ne comportent généralement pas d'informations utiles au-delà de 1kHz ; cela permet de limiter le cadencement du SoC jusqu'à une gamme 10-100kHz, motivant alors un abaissement de la tension d'alimentation vers le régime de sous-seuil. Forte d'un accélérateur HW FFT performant, d'une cellule SRAM optimisée pour opérer en régime de sous-seuil, et d'un convertisseur DC-DC à très haut rendement permettant de faire varier la tension d'alimentation, le système est utilisé pour réaliser la détection du déclenchement des attaques épileptiques. Toutefois, les auteurs s'attardent essentiellement sur les aspects énergétiques de l'extraction des *features* : ils comparent notamment la détection en réalisant la FFT via l'accélérateur HW ou de manière SW sur le Cortex-M3. Par contre, aucun résultat concernant les performances de détection ne sont réellement fournit. Or, l'algorithme qu'ils utilisent repose sur un modèle simpliste à seuil pré-déterminé et n'apparaît donc pas comme étant très robuste [21], particulièrement vu l'absence de résultats exhaustifs.

La plate-forme proposée par **Kwong et al.** [32] est, elle, construite autour d'un GPP RISC 16-bit de type MSP430. Elle comporte notamment quatre accélérateurs différents relevés comme étant pertinents par les auteurs en raison de leur caractère commun à plusieurs algorithmes biomédicaux différents : un filtre à réponse impulsionnelle finie (FIR - suppression du bruit), un module FFT (analyse du contenu fréquentiel), filtre médian (suppression de pointes de bruit) et module CORDIC (fonctions mathématiques non-linéaires). Par le biais d'un travail d'optimisation important dans l'implémentation de ces accélérateurs, les auteurs réalisent des gains énergétiques allant jusqu'à un facteur 215 par rapport à l'exécution SW. Par ailleurs, ils démontrent leur utilisation dans la partie FE de deux algorithmes biomédicaux (classification non-réalisée) : d'une part pour la détection du déclenchement des crises d'épilepsie via EEG, et d'autre part pour la détection d'arythmie via signaux électrocardiogrammes (ECG). La réduction en consommation par rapport à une réalisation entièrement SW sur le GPP de ces algorithmes est respectivement d'un facteur 10.2 et 11.5.

K.H. Lee et Verma [33] établissent le même genre de SoC autour d'un MSP430 et pourvu de différents accélérateurs HW dans le but de pouvoir implémenter plusieurs algorithmes biomédicaux. Toutefois, et à l'instar des solutions entièrement dédiées évoquées précédemment, ils utilisent des modèles basés sur les données, ce qui motive l'implémentation de fonctions discriminatives de *machine-*

		Unités	Sridhara [31] **JSSC11**	Kwong [32] **JSSC11**	Lee [34] **VLSI13**	Shoaib [35] **TVLSI13**
Technologie (CMOS)		-	130nm	130nm	130nm	150nm FD-SOI
Processeur GP		-	ARM Cortex-M3	MSP430	MSP430	Tensilica
Algorithme implémenté [a]		-	Seuil [26]	Shoeb [17]	Shoeb [11]	-
Base de données		-	Freiburg [20]	CHB-MIT [19]		-
Accél. HW	**FE**	-	FFT	Multiples [b]	CORDIC/MAXMIN	-
	Class.	-	-	-	Configurable [c]	SVM
Noyau SVM		-	-	-	Configurable	Polynomial
Surface de puce		mm^2	8.88	12	7.49	2.9
Cadencement		MHz	0.007-5	-	-	0.55/2/10
Tension d'alimentation		V	0.5-1	0.5-1	0.7-1.2	0.4/0.6/1.2
Energie [d]	**SW**	μJ/FV	-	198	4053	-
	HW		-	19.3	93.6	-

a. Nous considérons uniquement l'algorithme implémenté pour la détection du déclenchement des crises d'épilepsie, et pas les éventuels autres algorithmes (Arythmie via ECG, etc.)

b. FIR, filtre médian, FFT, CORDIC.

c. Supporte de nombreuses techniques de *machine-learning* : SVM, AL, GMM, HMM, PCA, etc.

d. Considérations énergétiques sur le traitement complet par le SoC d'un vecteur de *features* (jusqu'à classification si présente) lorsque l'algorithme est réalisé soit complètement en SW, soit avec l'utilisation d'accélérateurs HW. Les solutions [31] et [35] n'implémentant le même algorithme que celui étudié dans ce travail, les données énergétiques ne sont pas reprises : le premier parce que les performances du détecteur ne sont pas fournies, et le deuxième parce que l'objectif de détection est différent (arythmie via signaux ECG).

Tableau 1.2 – Comparaison de plate-formes SoC biomédicales réalisant notamment (ou potentiellement) la détection d'attaques épileptiques

learning. Le constat est donc différent à deux égards. Tout d'abord, les fonctions de FE seront plus variables d'un algorithme à l'autre que le module de classification qui peut-être le même (p.ex. SVM). Ensuite, pour un algorithme donné, la réalisation en SW sur un GPP du FE et de la classification montre que la consommation liée à cette dernière est proportionnelle à la complexité du modèle, là où celle liée au FE reste constante. En conclusion, la première devient rapidement plus importante que la seconde, et la réalisation d'accélérateurs HW pour celle-ci se justifie alors nettement plus ; deux sont implémentés dans ce SoC. Premièrement, un accélérateur SVM (SVMA) configurable permet d'utiliser différents noyaux (linéaire, polynomial, gaussien), selon la complexité requise par l'application et afin de limiter au mieux la consommation. D'autre part, un module dédié d'apprentissage actif (AL) permet de réaliser un système en boucle fermée : les données sEEG acquises sont triées et renvoyées pour évaluation par des experts, de façon à faire évoluer le modèle avec le temps et donc la précision du classificateur. A l'instar de Kwong et al. [32], l'épilepsie (via sEEG) et l'arythmie (via ECG) sont traitées par la plate-forme mais avec, dès lors, la classification en plus : les économies d'énergie par rapport à une exécution SW n'en sont que plus importantes, respectivement d'un facteur 62 et 145.

Par ailleurs, dans [34], **K.H. Lee et Verma** étendent largement leur plate-forme et de façon très modulaire, leur permettant de supporter de nombreuses techniques de *machine-learning* : les SVM

(régression et classification), l'AL, les modèles de mélanges gaussiens (GMM), les modèles de Markov cachés (HMM), l'analyse en composantes principales (PCA), etc. En outre, cela leur permet d'avoir également des accélérateurs HW configurables pour des opérations clé de FE, comme les filtres FIR ou encore les transformées en ondelettes discrètes. Le tout est rendu possible par un jeu de machines à états finis (FSM) et un contrôleur de noyaux pilotant trois modules arithémiques : un CORDIC (fonctions non-linéaires), un MAXMIN (recherche d'un index ou d'un max./min. dans un tableau) et surtout d'une unité de calcul (DCU) très modulable (sommes, multiplications, décalages, etc.). Les auteurs démontrent l'efficacité de leur SoC dans le cadre de six algorithmes différents, dont plusieurs de détection d'arythmie par signaux ECG, mais également la détection du déclenchement des attaques épileptiques par signaux sEEG (algorithme de Shoeb [11]).

Enfin, **Shoaib et al** [35] rapportent également une plate-forme SoC biomédicale autour d'un GPP Xtensa Tensilica dont l'attrait est la facilité d'ajout d'instructions customisées (CI). Cela permet aux auteurs une démarche en trois niveaux d'implémentation : complètement logicielle sur le GPP, puis en ajoutant des CI, et enfin avec un coprocesseur dédié (SVM). Il en ressort que les CI sont très appropriées - d'un point de vue de la réduction en consommation - pour les opérations de FE, mais beaucoup moins pour la classification en raison de la quantité de données importantes manipulées. En outre, et à l'instar de [33], les auteurs mettent en avant le besoin de flexibilité nettement plus important dans les opérations de FE lorsque l'on souhaite supporter un certain nombre d'algorithmes biomédicaux, là où la classification peut-être réalisée la plupart du temps via une SVM. Ces deux considérations justifient l'ajout du coprocesseur SVM dédié, avec un noyau polynomial configurable. Les auteurs ne présentent pas de résultats exhaustifs d'implémentation de différents algorithmes. Toutefois, ils réalisent une étude énergétique approfondie des différents niveaux d'implémentation possibles dans leur système pour la détection d'arythmie, et en faisant varier différents paramètres : la complexité du noyau, la tension d'alimentation ou encore la précision de représentation des données. Par exemple, en ce qui concerne l'énergie par classification (à noyau SVM fixe et en comptant les opérations de FE) les auteurs démontrent une réduction de consommation d'un facteur 2 en utilisant les CI, d'un facteur 228 en ajoutant le coprocesseur dédié (V_{DD} à 1.2V), d'un facteur 273 en diminuant alors la précision des données de 12-bit à 8-bit, et enfin d'un facteur 1170 en réduisant de surcroît la tension d'alimentation de manière agressive (V_{DD} à 0.4V).

1.5 Synthèse

Ce chapitre nous a permis de poser le cadre de ce travail : le portage sur un SoC à faible consommation d'un algorithme de détection du déclenchement des attaques épileptiques sur base de signaux sEEG. Dans un premier temps, les motivations de développer un tel système ont été établies sur base du contexte de l'épilepsie ainsi que des différentes techniques possibles (types de détecteurs, types d'EEG). Ensuite, nous avons vu les différents aspects systèmes en partant des challenges de conception et en abordant de manière générale l'architecture d'un système performant. Ensuite, la

présentation de l'état-de-l'art dans le domaine a permis de confirmer la pertinence de l'utilisation d'un modèle basé sur les données ; de plus, un algorithme approprié a été mis en évidence (Shoeb [11]).

En outre, l'état-de-l'art montre deux voies possibles : celle des solutions entièrement dédicacées, plus performantes, et celle des plate-formes biomédicales, plus flexibles et modulaires. Ces deux approches présentent chacune un intérêt. Toutefois la démarche de ce travail s'inscrit plutôt dans la seconde : nous montrerons l'implémentation de l'algorithme de Shoeb, avec extraction des features en SW sur un GPP, accompagné d'un accélérateur HW réalisant la classification via une SVM. En effet, étant donnée l'évolution des microprocesseurs pour systèmes embarqués - tels que le ARM Cortex-M0 utilisé dans ce travail - privilégier la modularité dans un travail de recherche nous parait plus intéressant. A fortiori, les travaux [35] et [36] montrent l'intérêt de l'utilisation d'une SVM dans d'autres algorithmes de détection biomédicale, sur base de signaux ECG cette fois.

La suite de ce travail est organisée comme suit : le chapitre 2 présente et discute l'algorithme de Shoeb et pose les bases théoriques nécessaires à sa paramétrisation, ainsi qu'à l'implémentation de l'accélérateur SVM. Ensuite, l'implémentation SW est décrite et validée au chapitre 3 : d'abord via Matlab, en raison de la facilité de développement et de production de résultats exhaustifs validant l'implémentation de l'algorithme. Enfin, le chapitre 4 concerne la réalisation de la plate-forme SoC autour d'un microprocesseur ARM Cortex-M0, avec le portage du détecteur SW en C et la conception de l'accélérateur HW SVM.

Chapitre 2

Algorithme de détection d'attaque

L'objectif de ce chapitre est de présenter l'algorithme de détection de déclenchement des crises d'épilepsie implémenté. Dans un premier temps, nous l'introduirons de manière générale. Ensuite, nous présenterons les critères permettant de juger de la qualité de ce type d'algorithme afin de pouvoir pleinement comprendre le détail des deux composantes principales que nous présenterons ensuite : l'extraction des *features* et la classification.

2.1 Vue d'ensemble

L'algorithme implémenté dans ce travail [1] a été développé par Ali H. Shoeb, dans le cadre de sa thèse de doctorat au MIT [2] (2005-2009) [8] et notamment publié via [11]. Le but de cette partie est de présenter l'algorithme de manière générale.

Nous avons vu à la partie 1.3.2) l'intérêt de travailler avec des modèles basés sur les données et non sur la caractérisation des processus physiologiques complexes sous-jacents. La Figure 2.1 rappelle les deux aspects principaux d'un tel algorithme de détection biomédical : l'extraction de biomarqueurs pertinents (paramètres des signaux ayant une corrélation forte avec l'état physiologique cible du patient) et leur interprétation [21]. Et lorsque les biomarqueurs sont traités sous forme de vecteur de *features* (FV), l'interprétation peut être réalisée par un classificateur issu des techniques de *machine-learning*.

Cette modélisation sur base des données est l'un des attrait de l'algorithme choisi. Comme entrevu à la partie 1.2.4), le contenu spectral des signaux sEEG est un biomarqueur pertinent dans la détection du déclenchement des attaques épileptiques. Le principe développé par Shoeb est de construire un FV en combinant ces *features* dits "spectraux" avec deux autres types : spatiaux et temporels. Nous verrons à la partie 2.3 comment cette association est réalisée, et pourquoi certaines caractéristiques inhérentes aux signaux sEEG (partie 1.2.3) la justifient.

L'objectif ultime de l'algorithme est d'identifier les signaux sEEG entrants comme étant ictaux (état de "crise") ou interictaux (état normal [3]). Cette classification binaire est donc réalisée par une technique de *machine-learning* : les machines à vecteurs de support (SVM). Sur base d'un ensemble

1. Etant donné que le présent travail n'a pas été réalisé au niveau de l'algorithme, le contenu des parties 2.3 et 2.4 (dont certaines figures) a été partiellement adapté directement de [8].
2. Massachussets Institute of Technology, Boston, Masschussets, USA.
3. NB : nous qualifions donc d'état normal tout état physiologique ne relevant pas d'une crise épileptique ; toutefois, nous avons vu à la partie 1.3.1 que les signaux sEEG mettent en évidence d'autres états physiologiques anormaux que ceux liés à l'épilepsie.

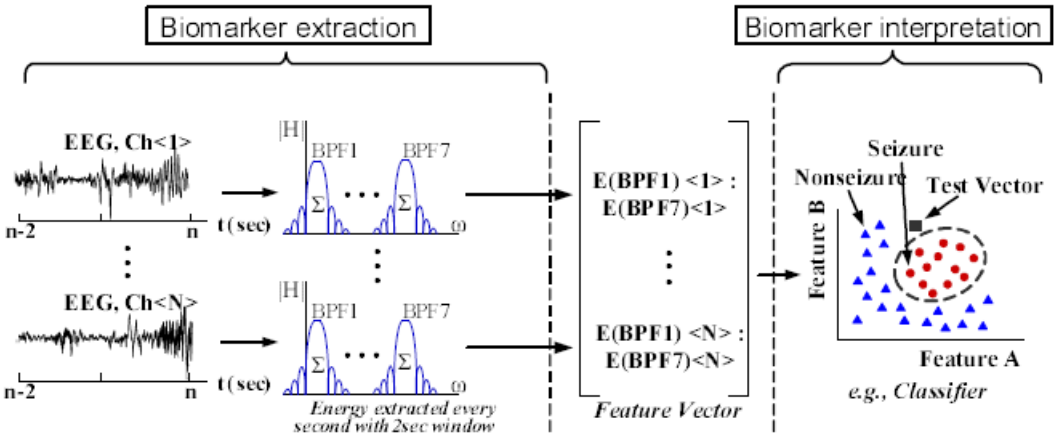


Figure 2.1 – Structure typique d'un algorithme de détection biomédical basé sur les données appliqué à l'épilepsie et sur base de signaux sEEG [21]

de données dit "d'entraînement", cette SVM permet de construire un modèle capable d'établir, selon un certain degré de précision, la corrélation entre les biomarqueurs et les états physiologiques cibles du patient. Le succès du classificateur dépend de plusieurs facteurs, dont la pertinence du set de données et des *features* extraits, la complexité du modèle utilisé, etc., qui seront discutés à la partie 2.4.

Nous avons vu dans l'état-de-l'art des IC pour la détection du déclenchement d'attaques épileptiques sur base de signaux sEEG (partie 1.4) que la plupart des solutions existantes implémentent l'algorithme de Shoeb. Il s'agit là d'une indication forte sur sa pertinence, confirmée par l'analyse de l'état de l'art du domaine. En effet, il réalise à notre sens le meilleur compromis entre simplicité des *features* extraits et bonnes performances de détection (voir chapitre 3), ce qui en fait le candidat idéal pour un portage sur un SoC basse-consommation. Pour des revues plus exhaustives sur le sujet (détecteurs d'événement, signaux iEEG, utilisation complémentaire d'autres signaux physiologiques), le lecteur est renvoyé vers les articles [24] (2012) et [25] (2013).

2.2 Métriques de performances

De manière générale, on parle de résultat positif lorsqu'un algorithme réalise une détection. Par conséquent, les "vrais positifs" (TP) traduisent une détection correcte d'attaque épileptique, à l'inverse des "faux positifs" (FP). D'un autre coté, les "vrais négatifs" (TN) réfèrent logiquement aux signaux normaux (hors état de crise) correctement classés, tandis que les "faux négatifs" (FN) désignent les signaux anormaux non-relevés par le détecteur. Il est à noter que les données d'entraînement et de test (présentées plus en détail à la partie 3.1)sont labellisées au préalable par un expert électroencéphalographiste .

A partir de cette nomenclature, nous pouvons établir plusieurs métriques de performances pour notre détecteur. Cependant, il est important de noter que leur définition peut différer selon qu'il

s'agisse d'un détecteur de déclenchement ou d'événement (partie 1.2.2). En effet, étant donné que les algorithmes segmentent - d'un point de vue temporel - les signaux EEG avant de les traiter, il est possible de juger de la qualité du détecteur en considérant le classement de chaque segment. Toutefois, l'objectif d'un détecteur déclenchement ne réside pas dans l'exactitude de labellisation de chaque segment (à l'inverse de ceux dits "d'événements"), mais dans le fait d'identifier le plus de crises possibles, en minimisant le délai de détection et le nombre de fausses alarmes.

Voici les métriques couramment utilisés dans la littérature pour caractériser les performances des détecteurs de crise d'épilepsie [9], [14] ; lorsque c'est nécessaire, nous précisons la définition de la métrique correspondant pour les détecteurs d'événements, que nous renommons afin de pouvoir les différencier par la suite :

- **Latence** : Lat, représente le délai en secondes entre le déclenchement électrique d'une attaque épileptique (voir partie 1.2.4), et sa détection.

- **Sensibilité** : caractérise la propension du détecteur à identifier une crise. Dans le cas des détecteurs de déclenchement (SeD), il s'agit du pourcentage d'attaques soumises au classificateur qui sont détectées. Par ailleurs, dans le cas des détecteurs d'événements (SeE), il s'agit du pourcentage de "vrais positifs", calculé par le rapport entre les segments TP et la somme des segments d'attaques soumis au détecteur :

$$SeE = \frac{TP}{TP + FN} \times 100 \tag{2.1}$$

- **Spécificité** : lorsque l'objectif de détection est le déclenchement des crises (SpD), il s'agit du nombre moyen de fausses alarmes par heure. Pour la détection dite d'événements épileptiques, ce critère (SpE) fait état de la capacité à identifier les états normaux du patient, soit le taux de "vrais négatifs", exprimé par le pourcentage suivant :

$$SpE = \frac{TN}{TN + FP} \times 100 \tag{2.2}$$

- **Précision** : Pr, généralement utilisé pour les détecteurs d'événements, il s'agit du pourcentage de segments correctement classés, par rapport à l'ensemble des segments soumis au détecteur :

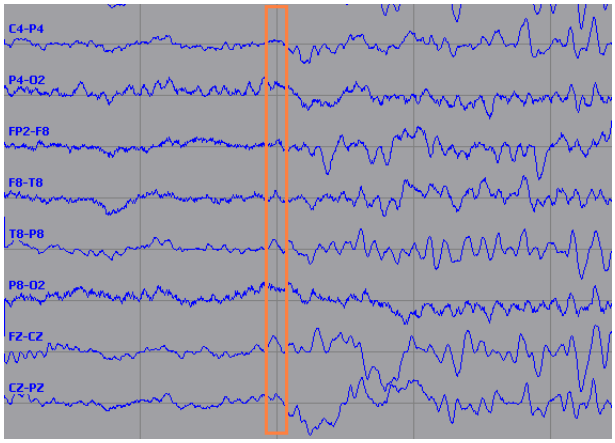
$$\mathrm{Pr} = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \tag{2.3}$$

2.3 Construction du vecteur de features

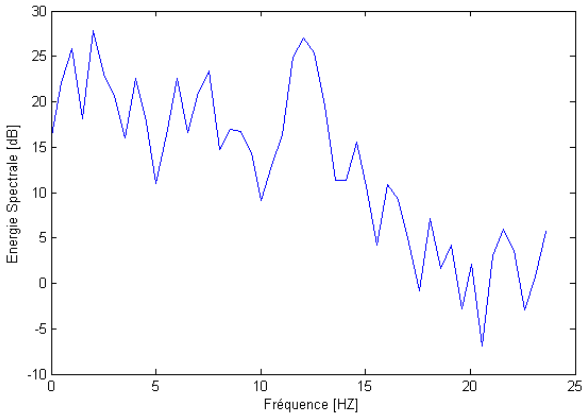
Le principe sur lequel repose l'algorithme de détection a été abordé à la partie 1.2.4 : le déclenchement des attaques présente une activité rythmique notable sur un ou plusieurs canaux sEEG, ce qui induit un changement soudain dans la distribution de l'énergie spectrale de ceux-ci. Nous allons voir ici la manière dont sont extraits les *features*, mais aussi, comment et pourquoi ils sont combinés selon des considérations spatiales et temporelles dans un seul et même vecteur.

2.3.1 Features spectraux

L'activité rythmique développée lors du déclenchement d'une crise d'épilepsie peut être composée de différentes composantes fréquentielles, variables selon différents facteurs possibles, dont le type d'attaque, son origine dans le cerveau, l'âge, l'état de conscience, etc. La Figure 2.2a donne un exemple de déclenchement électrographique d'une attaque par le biais d'une partie des canaux sEEG d'un patient ; en outre le spectre du canal FP2-F8 au moment du déclenchement et pour une fenêtre temporelle de deux secondes est donné à la figure 2.2b.



(a)



(b)

Figure 2.2 – Exemple de déclenchement électrographique d'une attaque épileptique : (a) un sous-set des canaux sEEG (extraits via EDFbrowser) et (b) énergie spectrale du canal FP2-F8 pour une fenêtre temporelle de 2 secondes mettant en évidence plusieurs composantes fréquentielles dominantes.

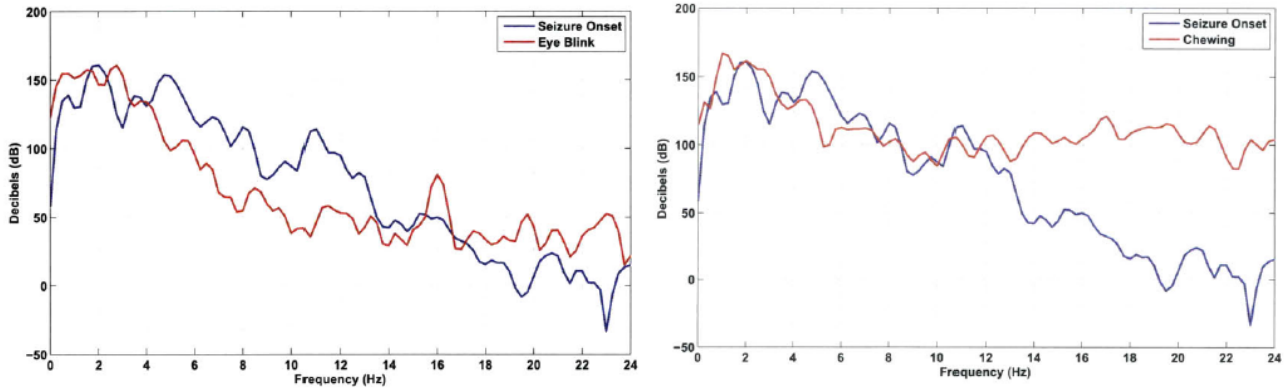
Il est à noter que prendre en compte uniquement la composante fréquentielle d'énergie maximale n'est pas suffisant si l'on souhaite optimiser la spécificité du détecteur (éviter les fausses alertes). En effet, il existe d'autres types d'événements chez le patient qui induisent une activité rythmique notable et pour lesquelles la composante fréquentielle principale peut coïncider avec celle du déclenchement d'une attaque.

Les Figures 2.3a et 2.3b illustrent cela, en comparant la distribution de l'énergie spectrale [4] pour le même canal sEEG d'un patient lors du déclenchement d'une attaque et lors de deux autres événements :

4. Calculée sur une fenêtre temporelle de deux secondes.

respectivement une série rapide de clignement d'yeux, et le fait de mâcher un *chewing-gum*.

Dans les deux cas, on peut observer des chevauchements sur une certaine gamme de fréquence qui rendraient la différenciation des états physiologiques difficile en se basant uniquement sur la composante fréquentielle principale. Ces figures mettent par ailleurs en évidence les divergences de spectre notables qui permettent, en pratique, cette différenciation.



(a) (b)

FIGURE 2.3 – Comparaison de l'énergie spectrale du signal sEEG (FP2-F4) d'un patient épileptique lors du déclenchement d'une attaque (en bleu) avec deux autres types d'événements induisant une activité rythmique (en rouge) : (a) une série rapide de clignements d'yeux et (b) le fait de mâcher un *chewing-gum* [8]

La gamme de fréquence pertinente dans le cas d'une mesure de l'activité cérébrale par sEEG est 0.5-25 Hz. Pour rappel, les fréquences supérieures sont atténuées notamment par le squelette et le liquide cérébrospinal, ce qui n'est pas le cas pour les signaux iEEG (0.5-105 Hz) [13].

Les *features* spectraux sont extraits comme suit : le spectre de chaque canal EEG k est calculé sur une période de L secondes avant d'être divisée en M sous-bandes S_i (avec i = 1,...,M) pour lesquelles l'énergie spectrale $x_{i,k}$ est calculée (voir figure 2.4).

2.3.2 Features spatiaux et temporels

La distribution spatiale (canaux EEG impliqués parmi les N disponibles) est un autre élément important dans la différenciation des événements induisant une rythmique notable dans l'activité cérébrale mesurée. A plus forte raison, son utilisation en complément des *features* spectraux décrits précédemment permet d'améliorer encore la spécificité du détecteur. La Figure 2.5 reprend la construction du vecteur $\boldsymbol{X}_T$ $(M \times N)$ combinant l'information spectrale et spatiale contenue sur une période de L secondes d'EEG à un instant $t = T$.

Enfin, il est important de caractériser la façon dont le vecteur courant $\boldsymbol{X}_T$ est influencé par son passé proche. En effet, de cette façon, le modèle sera capable de déterminer comment le déclenchement d'une attaque émerge depuis des signaux sEEG "normaux", ainsi que l'évolution des signaux pendant

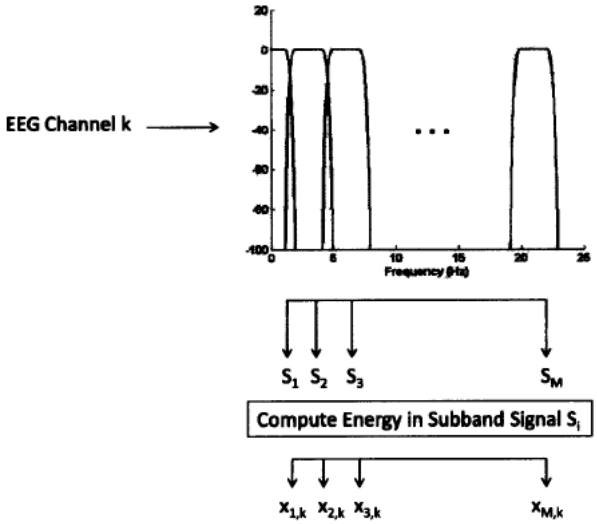


FIGURE 2.4 – Procédé de calcul des *features* spectraux [8]

ce déclenchement.

Dans cette optique, une dernière étape dans la construction du vecteur de *features* consiste à empiler une série de W vecteurs $\boldsymbol{X}_T$ caractérisant plusieurs périodes de L secondes contiguës, mais sans chevauchement [5] (voir Fig. 2.5). Il est important de noter que ce procédé ne produit pas le même résultat que le fait de former un vecteur $\boldsymbol{X}_T$ sur une période L plus longue (d'un facteur W). De fait, dans le premier cas, les événements discrets sont préservés ; $W \times M \times N$ *features* sont fournis au classificateur là où il y en aurait $M \times N$ dans l'autre cas.

2.3.3 Discussion des paramètres

- **Longueur des fenêtres temporelles EEG : L.** Nous avons abordé et illustré précédemment (partie 1.3.1) le fait que les signaux sEEG sont non-stationnaires. De ce fait, il est important d'extraire les *features* spectraux sur une période de temps raisonnablement courte. Toutefois, pour que cela reste pertinent d'un point de vue physiologique, une durée L de 2 secondes est typiquement utilisée.

- **Nombre de canaux EEG : N.** Le nombre d'électrodes est typiquement de 22 (système international "10-20", voir partie 1.2.3), formant 18 canaux. L'algorithme originel de Shoeb les utilise tous, soit $N = 18$. Il est intuitivement évident que cela permet de maximiser les performances du détecteur lui-même.

  Toutefois, Shih et al [38] ont développé un algorithme de sélection des *features* basé sur la réduction du nombre de canaux : leurs résultats montrent que l'on peut descendre jusqu'à $N = 5$ en gardant des performances décentes. En effet, en appliquant leur algorithme sélectif à celui

5. Non justifié par l'auteur.

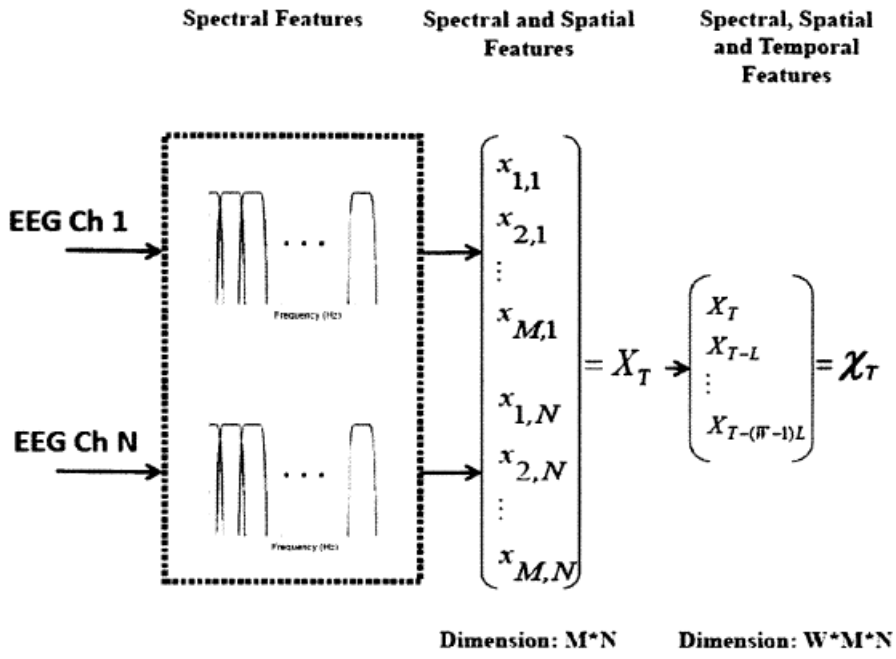


FIGURE 2.5 – Illustration des différentes étapes de la construction du vecteur de *features* : (1) extraction de l'énergie spectrale en divisant le spectre des N canaux sEEG en M sous-bandes, (2) formation d'un vecteur $\boldsymbol{X}_T$ concaténant les N canaux afin d'ajouter l'information spatiale, (3) formation du vecteur final $\boldsymbol{\mathcal{X}}_T$ en empilant W vecteurs $\boldsymbol{X}_T$ afin de capturer l'information temporelle. [13]

de détection développé dans [17], leur détecteur perd 2% de sensibilité, nécessite 2s de latence supplémentaire, mais réduit, par contre, le taux moyen de fausses alarmes par heure d'un facteur 2. En outre, ils réalisent un portage SW sur un système ambulatoire équipé d'un processeur de traitement de signaux numériques (DSP) à usage "général". En effet, l'un des attraits principaux de cette technique est de diminuer la consommation d'un tel système, et donc la durée de vie de la batterie. En l'occurrence, ils obtiennent une réduction de 69% de la consommation par rapport au portage sur un système similaire de l'algorithme de détection sans la sélection des canaux [37].

- **Nombre de sous-bandes par spectre : M.** Shoeb caractérise les performances de son algorithme en faisant varier le nombre de sous-bandes M (2,4,8) par lequel le spectre de chaque fenêtres sEEG est divisé (W fixé à 3). Il a pu ainsi montré que l'augmentation de ce paramètre impacte de manière négligeable la latence ou la sensibilité du détecteur. Par contre, cela réduit le nombre de fausses alarmes d'un facteur 3 (en passant de M=2 et M=8 (valeur retenue)).

- **Nombre de vecteurs intermédiaires dans $\boldsymbol{\mathcal{X}}_T$: W.** A l'instar du paramètre M, Shoeb optimise son détecteur en regard du nombre W de vecteurs intermédiaires empilés dans le vecteur de *features* complet $\boldsymbol{\mathcal{X}}_T$ (pour M fixé à 8). A nouveau, la sensibilité du détecteur en reste inchangée ; W est fixé à 3, soit le meilleur compromis entre la latence et la spécificité.

2.4 Classification

L'objectif de l'algorithme implémenté est de classer les signaux sEEG mesurés, de manière binaire : état de crise épileptique (classe C_1, label "+1") et état normal (classe C_2, label "-1"). A cet effet, la partie 2.3 a montré comment ces signaux peuvent être caractérisés par un vecteur de *features* pertinents $\boldsymbol{\mathcal{X}}_T$. Le but de cette partie est de montrer comment il est possible de construire une fonction de labellisation $f(\boldsymbol{\mathcal{X}}_T)$ attribuant un label $y = \pm 1$ au vecteur $\boldsymbol{\mathcal{X}}_T$.

L'outil utilisé pour réaliser cette classification est une machine à vecteur de support (SVM). L'objectif ici est de développer le principe de la SVM ainsi que son intégration dans l'algorithme implémenté. Enfin, nous reprendrons dans une dernière partie les différents paramètres de l'algorithme relatifs à la classification et répondant au mieux aux contraintes imposées par notre problème de détection.

2.4.1 Machines à vecteurs de support (SVM)

Les machines à vecteurs de support (SVM) sont une technique de *machine-learning* qui s'est principalement développée [6] depuis le milieu des années 90 et dont l'un des pionniers majeurs est V. Vapnik [39,40]. Elle ont été fortement popularisées depuis, en raison de leur très bonnes performances, particulièrement lorsque les sets de données sont à haute dimension et fortement déséquilibrés (une des classes peu représentée) [8]. Nous allons ici en approfondir le principe [41–43]. Il est à noter que le but n'est pas de fournir une description mathématique exhaustive [7] mais de poser les bases théoriques nécessaires à la compréhension du choix de modèle et des paramètres ainsi que, par la suite, de l'implémentation HW du classificateur SVM.

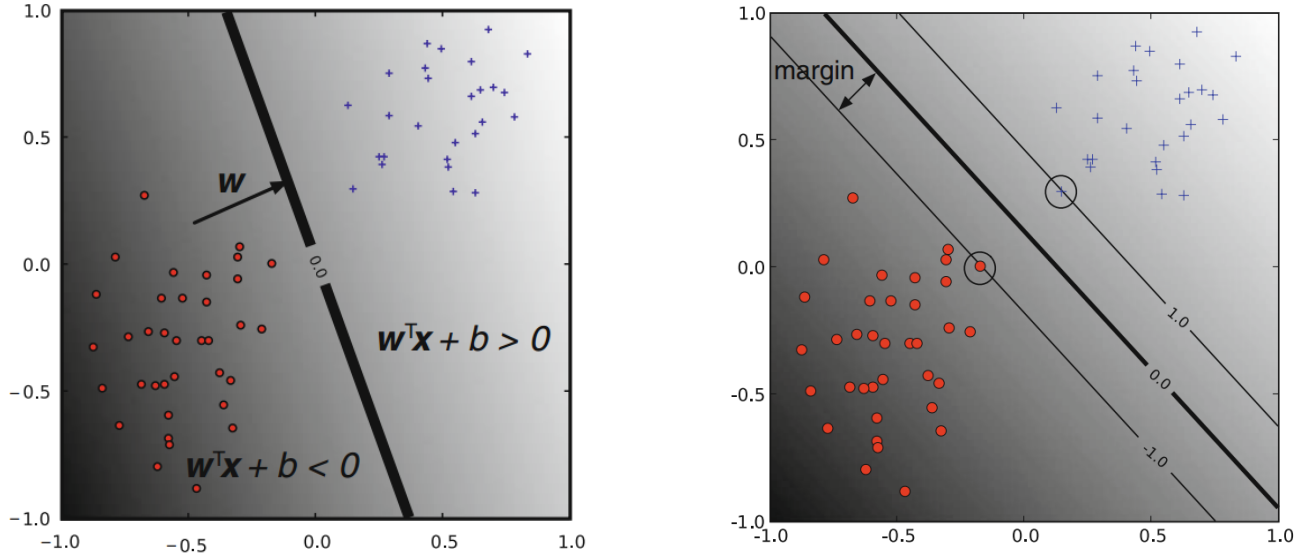
Principe général et notations. Les SVM sont une méthode de classification discriminante à apprentissage supervisé. Les approches de classification discriminante sont basées sur la création d'une frontière de décision divisant l'espace des *features* [21], ici en deux classes [8] : C_1 (label y=+1) et C_2 (label y=-1). Lorsque cet espace est à 2 dimensions (2-D), cette séparation est dès lors simplement une ligne (voir Fig. 2.6a) ; de manière générale, elle est qualifiée d'hyperplan. Bien que le problème que nous traitons requiert un nombre relativement important de *features* [9], cette représentation 2-D permet de faciliter le développement. Néanmoins, ce qui suit est directement transposable à D dimensions.

6. NB : les premiers travaux théoriques de Vapnik remontent toutefois aux années 60.
7. A cet effet, le lecteur est renvoyé vers [43] ou [44].
8. De base, les SVM sont des classificateurs binaires [42] ; elles peuvent être généralisées pour des problèmes multi-classes, mais étant donné que ce n'est pas notre cas, nous n'en ferons pas état ici.
9. cf. partie 2.3.3 : le vecteur de *features* est de dimension $W \times M \times N$, soit selon les paramètres fixés : $3 \times 8 \times 23 = 552$

Outre la notion de classificateur discriminant, les SVM sont dites à apprentissage supervisé. Cela requiert deux sets de données, l'un d'entraînement ($\boldsymbol{S}_E$) et l'autre de test ($\boldsymbol{S}_T$), composés respectivement de I et J couples "vecteur de *features* - label" $(\boldsymbol{X}, y)$:

$$\begin{cases} \boldsymbol{S}_E = \{(\boldsymbol{X}_1, y_1), (\boldsymbol{X}_2, y_2), ..., (\boldsymbol{X}_I, y_I)\} \\ \boldsymbol{S}_T = \{(\boldsymbol{X}_1, y_1), (\boldsymbol{X}_2, y_2), ..., (\boldsymbol{X}_J, y_J)\} \end{cases} \quad \text{avec} \quad \boldsymbol{X}_i, \boldsymbol{X}_j \in \Re^2 \quad \text{et} \quad y_i, y_j \in \{+1, -1\} \tag{2.4}$$

L'idée générale de l'apprentissage supervisé (et donc des SVM) est d'utiliser le set $\boldsymbol{S}_E$ pour calculer le modèle - composé pour le moment uniquement de l'hyperplan - (phase d'entraînement) utilisé ensuite pour classer les données du set $\boldsymbol{S}_T$ (phase de test).



(a) (b)

FIGURE 2.6 – Une SVM linéaire (données 2-D, les vecteurs de *features* sont des points) : (a) dans sa forme la plus simple, la frontière de décision est un hyperplan (ligne en 2-D) selon lequel une fonction de discrimination g classe les nouveaux points (b) en pratique, l'hyperplan de séparation donnant lieu à la meilleure classification est déterminé par maximisation de la marge (distance entre l'hyperplan de séparation et les vecteurs de supports de chaque classe - points limites, encerclés), ce qui donne lieu à deux nouveaux hyperplans qui sont alors utilisés pour la classification [42]

Un classificateur linéaire. Dans sa forme la plus simple, une SVM est un classificateur linéaire basé sur une fonction g de discrimination (linéaire) exprimée par :

$$g(\boldsymbol{X}) = \sum_{i}^{I} w_i x_i + b = \boldsymbol{W}^T \cdot \boldsymbol{X} + b \tag{2.5}$$

avec $\boldsymbol{W}$ le vecteur de poids, et b le biais. Il s'agit donc d'une somme pondérée et biaisée. Si l'on reprend la Figure 2.6a (cas 2-D), notre hyperplan est l'ensemble des points $\boldsymbol{X}$ tels que $g(\boldsymbol{X}) = 0$. Etant donné la définition du produit scalaire (entre les vecteurs $\boldsymbol{W}$ et $\boldsymbol{X}$), l'hyperplan est ici la ligne

perpendiculaire à $\boldsymbol{W}$ et éloignée de l'origine par le biais, via $b/||\boldsymbol{W}||$.

Une fois cet hyperplan défini, les données de test peuvent être classées via la fonction de labellisation suivante :

$$f(\boldsymbol{X}) = y = \begin{cases} +1 & \text{si} \quad g(\boldsymbol{X}) = \boldsymbol{W}^T \cdot \boldsymbol{X} + b \geq 0 \\ -1 & \text{si} \quad g(\boldsymbol{X}) = \boldsymbol{W}^T \cdot \boldsymbol{X} + b \leq 0 \end{cases} \tag{2.6}$$

Calcul du modèle : maximisation de la marge. Le modèle - pour l'instant constitué du vecteur de poids $\boldsymbol{W}$ et du biais b - est donc calculé durant une phase d'entraînement pour laquelle il est impératif que le set S_E contienne des instances des deux classes. Nous pouvons observer à la Figure 2.6a que plus d'une séparation est possible. Le but de la phase d'entraînement de la SVM est donc de déterminer l'hyperplan offrant la meilleure classification possible.

Dans notre exemple 2-D, les données sont linéairement séparables (classificateur dit "*large-margin*"), la solution est unique : le modèle est calculé en maximisant la marge de classification (voir Fig. 2.6b), soit la distance géométrique entre l'hyperplan et le point de chaque classe le plus proche de celle-ci. Ces points - les cas limites de chaque classe - sont les "vecteurs de support" ; ils déterminent, de part et d'autre de la frontière, deux nouveaux hyperplans définis par $\boldsymbol{W}^T \cdot \boldsymbol{X} + b = \pm 1$ qui sont alors utilisés pour discriminer les nouveaux points.

La distance (marge) entre ces deux hyperplans est $(2/||\boldsymbol{W}||)$. Autrement dit, maximiser la marge revient à minimiser $||\boldsymbol{W}||$. Le problème d'optimisation contraint [10] peut donc être formulé de la façon suivante :

$$\min_{\boldsymbol{W},b} \quad ||\boldsymbol{W}||^2 \tag{2.7}$$

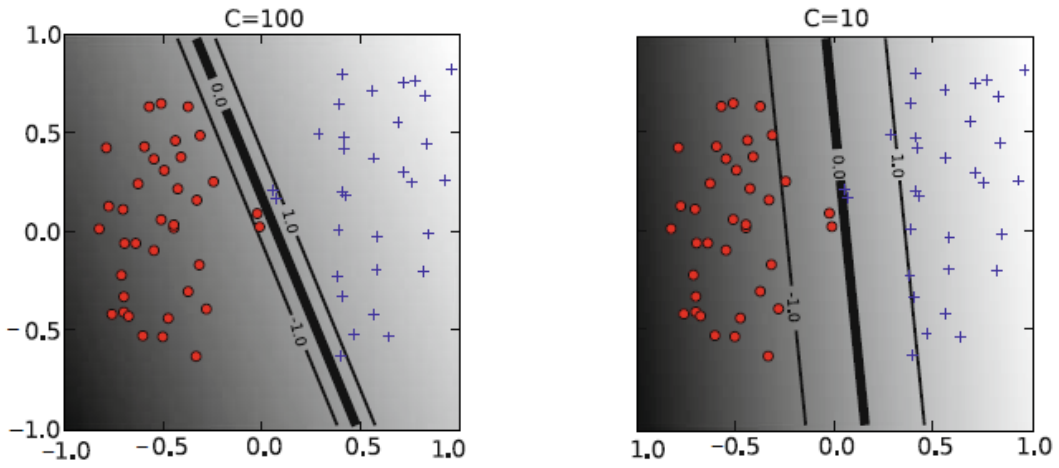
$$\text{t.q.} \quad y_i(\boldsymbol{W}^T \cdot \boldsymbol{X}_i + b) \geq 1, \forall \boldsymbol{X}_i \tag{2.8}$$

Données non-linéairement séparables : marge souple. Toutefois les données sont - en pratique, et particulièrement dans le milieu biomédical - rarement linéairement séparables. Pour gérer les points qui tomberaient au-delà des vecteurs de support de la classe à laquelle ils n'appartiennent pas ("*outliers*"), il est nécessaire d'ajouter à l'algorithme une marge souple ("*soft margin*"), sans quoi le problème d'optimisation n'aurait pas de solution.

Cette marge souple va permettre la construction du modèle malgré les *outliers*, mais au prix, bien entendu, des performances de classification. Elle constitue dès lors un compromis entre les violations

10. A noter que ce problème d'optimisation est généralement résolu via les multiplicateurs lagrangiens ; toutefois étant donné que notre implémentation HW se concentrera uniquement sur la phase de test (i.e. le modèle sera entraîné en SW), nous n'en ferons pas état ici.

de l'hyperplan "permises" (autrement dit, les erreurs de classification, que nous cherchons à minimiser), et la taille de la marge (que l'on cherche à maximiser). A ce titre, elle est paramétrable par l'utilisateur (facteur de marge souple C). La Figure 2.7 illustre cela, pour deux valeurs du paramètre C.



(a) (b)

FIGURE 2.7 – Illustration de l'effet du paramètre C (facteur de marge souple) sur la frontière de décision : une plus petite valeur implique une plus grande tolérance des "*outliers*" et donc une plus grande marge [42].

D'un point de vue mathématique, une variable ξ_i dite de "*slack*" est introduite pour chaque vecteur de *features*, correspondant à la distance entre un *outlier* $\boldsymbol{X}_i$ et l'hyperplan associé à sa classe. Le problème d'optimisation contraint devient alors :

$$\min_{\boldsymbol{W},b} \quad ||\boldsymbol{W}||^2 + C\sum_i \xi_i \tag{2.9}$$

$$\text{t.q.} \quad y_i(\boldsymbol{W}^T \cdot \boldsymbol{X}_i + b) \geq 1 - \xi_i, \forall \boldsymbol{X}_i \quad \text{avec} \quad \xi_i \geq 0 \tag{2.10}$$

L'algorithme tente de maintenir ξ_i à 0 (c'est-à-dire, de minimiser le *slack*), tout en maximisant la marge. On voit que le facteur "$C\sum_i \xi_i$" pénalise le modèle pour les instances mal classées ($\xi_i \geq 1$) ou se trouvant à l'intérieur de la marge ($1 > \xi_i > 0$). En d'autres termes, le paramètre C établit l'importance relative entre la maximisation de la marge et la minimisation du *slack* ; et lorsque $C \to \infty$, on se rapproche de la solution dite "à large marge" (voir Fig. 2.7a), étant donné que la tolérance des *outliers* est nulle.

Pour résumer, l'introduction de la marge souple permet de toujours avoir une solution au problème, mais requiert de la part de l'utilisateur de fixer le paramètre C. A noter que l'utilisation d'une marge souple peut aussi s'appliquer aux cas linéairement séparables, dans l'optique d'obtenir une marge plus importante.

Générer une séparation non-linéaire. Nous avons vu que l'utilisation d'une marge souple est nécessaire mais impacte les performances du classificateur. Afin de permettre à la SVM de mieux traiter les sets de données non-linéairement séparables [11], il est possible d'utiliser une fonction non-linéaire $\Phi(\boldsymbol{X})$ afin de projeter les données dans un espace de plus haute dimension où elles seront linéairement séparables (voir Fig. 2.8). En effet, il est possible de montrer, que lorsque le set de données est consistant [12], une telle fonction existe.

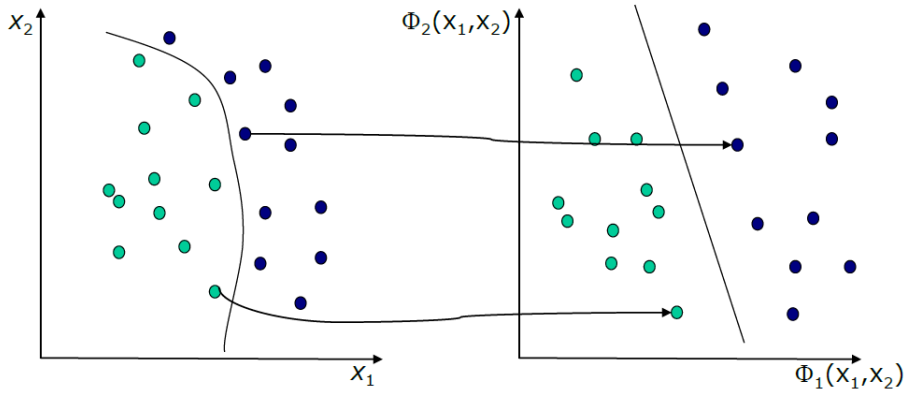


FIGURE 2.8 – Utilisation d'une fonction non-linéaire $\Phi(\boldsymbol{X})$ pour projeter les vecteurs de *features* dans un espace de dimension plus élevée où ils sont linéairement séparables [41]

La formulation du problème d'optimisation contraint devient :

$$\min_{\boldsymbol{W},b} \quad ||\boldsymbol{W}||^2 + C\sum_i \xi_i \tag{2.11}$$

$$\text{t.q.} \quad y_i(\boldsymbol{W}^T \cdot \Phi(\boldsymbol{X}_i) + b) \geq 1 - \xi_i, \forall \boldsymbol{X}_i \quad \text{avec} \quad \xi_i \geq 0 \tag{2.12}$$

En d'autres termes, la SVM détermine une frontière non-linéaire dans l'espace des *features* en résolvant une séparation linéaire dans un espace de plus haute dimension [8], c'est-à-dire en utilisant un classificateur linéaire. La question qui se pose est alors : pourquoi ne pas directement construire un classificateur non-linéaire ? L'un des arguments en faveur des classificateurs linéaires est la simplicité de l'algorithme d'entraînement, dont la complexité augmente plus lentement avec le nombre de données.

Formulation duale : l'astuce noyau. La détermination du modèle de la SVM est un problème d'optimisation quadratique sous contrainte affine. C'est ce qui le rend très efficace d'un point de vue calculatoire pour les systèmes modernes, du moins dans sa formulation la plus simple. En effet, le fait

11. Ce qui permet alors de diminuer le facteur de marge souple C
12. C'est-à-dire qu'il ne contient pas deux instances identiques avec des labels opposés

d'utiliser une projection non-linéaire et d'exprimer de manière explicite chaque vecteur de *features* dans le nouvel espace est généralement bien trop coûteux, particulièrement lorsque les vecteurs à l'entrée ont déjà une dimension importante.

L'idée de la méthode des noyaux est donc d'éviter cette correspondance explicite des données. A cet effet, le problème est réécrit sous sa forme duale, dite de Wolfe ; le vecteur de poids peut être exprimé comme une combinaison linéaire des données d'entraînement. Dès lors, la fonction de discrimination, avec l'utilisation d'une fonction non-linéaire, peut être exprimée par :

$$g(\boldsymbol{X}) = \boldsymbol{W}^T \cdot \Phi(\boldsymbol{X}) + b = \sum_{n=1}^{N_{SV}} \alpha_n \Phi(\boldsymbol{X}_n)^T \cdot \Phi(\boldsymbol{X}) + b = \sum_{n=1}^{N_{SV}} \alpha_n K(\boldsymbol{X_n}, \boldsymbol{X}) + b \tag{2.13}$$

avec N_{SV}, le nombre de vecteurs de support - soit les instances du set d'entraînement pour lesquelles le coefficient $\alpha_n > 0$, et K le noyau (ou fonction noyau). Or, il est possible de montrer qu'il existe des fonctions non-linéaires $\Phi(\boldsymbol{X})$ pour lesquelles le noyau K peut-être calculé sans devoir réaliser explicitement la correspondance des données dans le nouvel espace.

Choix du noyau. Les paragraphes précédents nous ont permis de mettre en évidence les limites de l'utilisation d'un noyau linéaire. Cette constatation s'applique à notre problème de détection, et aux algorithmes biomédicaux de manière générale [8]. En effet, nous avons vu dans l'état-de-l'art (partie 1.4.1) un exemple de système (Yoo et al. [29]) réalisant l'extraction des *features* selon l'algorithme de Shoeb (comme développé plus haut) et la classification via un noyau linéaire. En conséquence, les performances de détection en étaient fortement dégradées.

Dans la formulation de son algorithme [11], Shoeb a donc rapidement écarté l'utilisation d'un noyau linéaire, pour se tourner vers le noyau gaussien, appelé également noyau à fonction de base radiale (RBF). Celui-ci est l'un des plus couramment utilisés, en raison de son haut degré de flexibilité [45]. Toutefois, Shoeb se limitait à un développement SW et n'avait pas de contraintes énergétiques fortes. Le portage de son algorithme au niveau HW oblige à remettre ce choix en question.

Le Tableau 1.1 (chapitre 1) montre, via les résultats de Shoaib et al. [28], les différences énergétiques importantes entre l'utilisation d'un noyau linéaire, polynomial (degré 4) ou RBF. Ces résultats justifient en partie l'utilisation du noyau linéaire par Yoo et al. [29]. Toutefois la perte en sensibilité du détecteur n'est pas acceptable. Une possibilité pour améliorer ce critère de performances tout en gardant un noyau linéaire serait d'utiliser une variété nettement plus importante de *features* : le travail de Zabihi [14] met cela en lumière. Toutefois, là encore, pas de développement HW et pas de considérations énergétiques. Or, il est probable que l'utilisation d'un nombre encore plus important de *features* augmente nettement la consommation tant à l'extraction qu'à la classification, et ce, même en utilisant des techniques de réduction de données ("*features selection*").

Dès lors, étant donné le choix que nous avons porté de réaliser l'extraction des *features* sur un

microprocesseur et la classification via un accélérateur HW, deux constats peuvent être établis. Tout d'abord, l'utilisation de l'algorithme de Shoeb se justifie pleinement par son efficacité tout en ayant des *features* relativement peu complexes. Toutefois, cela implique que l'utilisation d'un noyau linéaire, qui nous arrangerait fortement d'un point de vue énergétique, n'est pas suffisante. Il est nécessaire de se tourner vers des noyaux non-linéaires. A cet effet nous considérerons les noyaux polynomial et gaussien. Le premier - comme nous le verrons au chapitre 4 - parce qu'il offre un bon compromis entre flexibilité et consommation, et le deuxième, parce qu'il s'agit d'un noyau performant et très couramment utilisé.

Enfin, voici les expressions des trois noyaux que nous utiliserons dans ce travail (avec $\boldsymbol{SV}_i$, le i-ème vecteur de support du modèle, et $\boldsymbol{X}$, le vecteur de *features* devant être classé) :

$$\textbf{Linéaire} : \qquad K(\boldsymbol{SV}_i, \boldsymbol{X}) = \boldsymbol{SV}_i \cdot \boldsymbol{X} \tag{2.14}$$

$$\textbf{Polynomial} : \qquad K(\boldsymbol{SV}_i, \boldsymbol{X}) = (\boldsymbol{SV}_i \cdot \boldsymbol{X} + 1)^d \tag{2.15}$$

$$\textbf{Gaussien} : \qquad K(\boldsymbol{SV}_i, \boldsymbol{X}) = \exp\left(-\gamma ||\boldsymbol{SV}_i - \boldsymbol{X}||^2\right) \tag{2.16}$$

La valeur des paramètres de modèle (appelés également hyperparamètres) d et γ sera discutée à la partie 2.4.3.

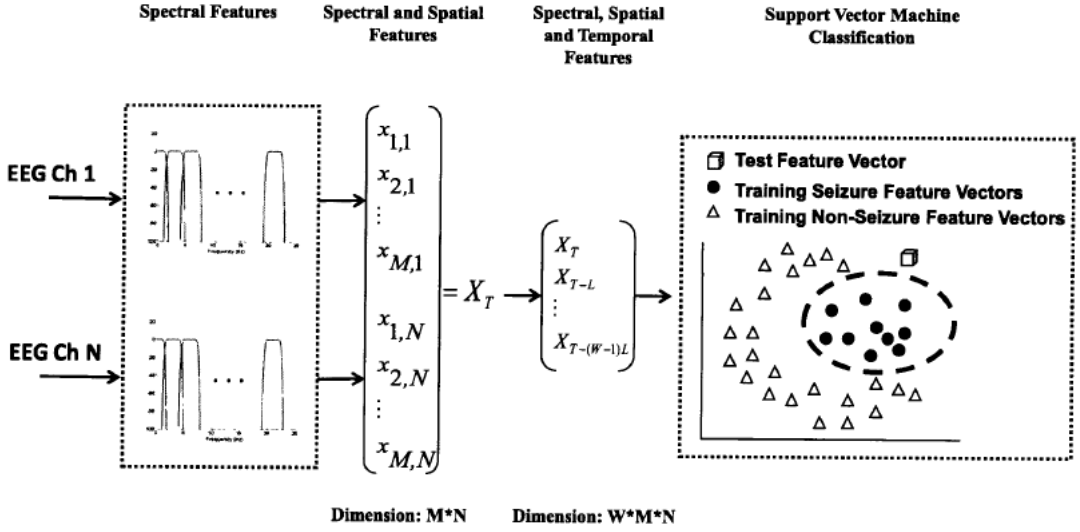


FIGURE 2.9 – Architecture complète de l'algorithme de Shoeb [11] de détection spécifique du déclenchement des crises d'épilepsie [8]

2.4.2 Intégration de la SVM dans l'algorithme

La Figure 2.9 reprend tous les éléments de l'algorithme. La SVM reçoit les vecteurs de *features* $\boldsymbol{\mathcal{X}}_T$ en entrée et leur attribue un label (± 1). Comme établi à la partie 1.3.1, la caractérisation des attaques via le type de *features* utilisés ici est sujette à une importante variabilité entre les individus. Dès lors afin d'optimiser la sensibilité du détecteur, il est important de traiter le problème de manière spécifique à chaque patient. Dans cette optique, la SVM doit être entraînée avec des FV issus de H heures de signaux sEEG enregistrés sur le patient traité. De plus, le set d'entraînement doit contenir des données issues des deux états physiologiques cibles : nous parlerons de vecteurs normaux et de vecteurs d'attaque.

La Figure 2.10a illustre la façon dont sont générés les vecteurs normaux. Nous avons fixé à la partie 2.3.3 les paramètres L à 2 et W à 3. Dès lors, chaque vecteur est formé de trois fenêtres distinctes de deux secondes. Par exemple, le vecteur complet $\boldsymbol{\mathcal{X}}_6$ (t = 6s) (figure 2.10a) est constitué des vecteurs intermédiaires $\boldsymbol{X}_2$, $\boldsymbol{X}_4$ et $\boldsymbol{X}_6$, et les deux derniers sont logiquement réutilisés pour le vecteur complet $\boldsymbol{\mathcal{X}}_8$ (en plus du vecteur intermédiaire $\boldsymbol{X}_8$), qui chevauche donc en partie $\boldsymbol{\mathcal{X}}_6$.

Le procédé est similaire pour générer les vecteurs d'attaque (figure 2.10b), à deux remarques près. Tout d'abord, le premier vecteur complet caractérisant l'attaque ($\boldsymbol{\mathcal{X}}_2$) est calculé en utilisant les vecteurs normaux $\boldsymbol{X}_{-2}$ et $\boldsymbol{X}_0$, de façon à préserver la continuité, ce qui permet au modèle "d'apprendre" comment une crise émerge d'un état normal. Ensuite, il est important de noter que toutes les données de l'attaque ne sont pas utilisées, mais uniquement les S premières secondes. La raison à ceci sera discuté à la partie suivante.

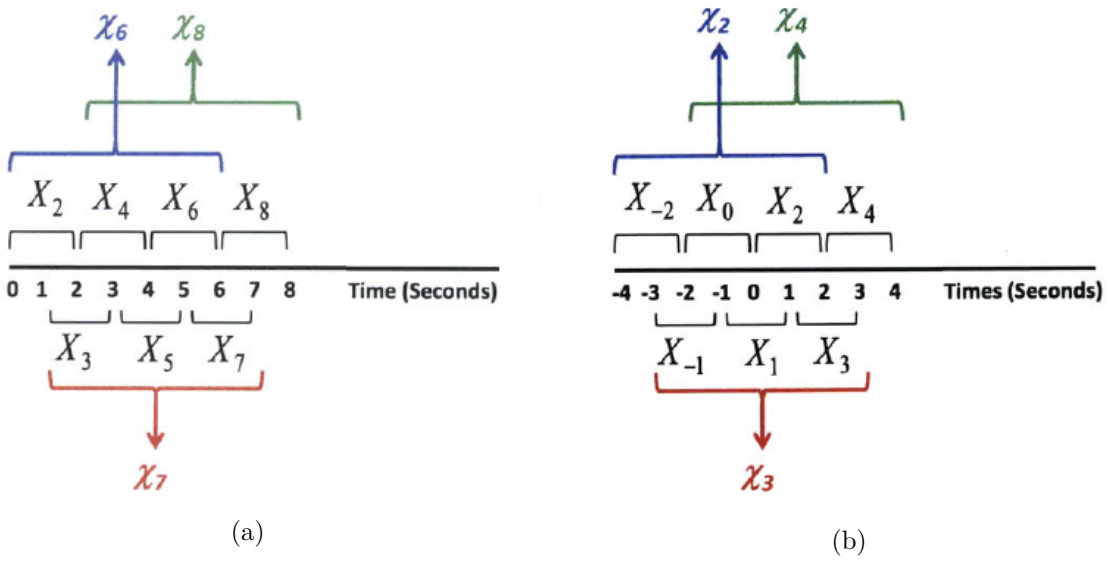


Figure 2.10 – Méthodologie de génération des vecteurs de *features* pour l'entraînement de la SVM : (a) vecteurs normaux (b) vecteurs d'attaque [8]

2.4.3 Discussion des paramètres

Les paramètres évoqués dans cette partie sont liés à la SVM en tant que telle (hyperparamètres), ainsi qu'à la phase d'entraînement de l'algorithme.

SVM : C, d et γ. L'effet du facteur de marge souple C a été développé à la partie 2.4.1 ; pour notre problème de détection, il a été optimisé par Shoeb à une valeur de $C = 1$ [17]. Les deux autres hyperparamètres sont relatifs au noyau utilisé ; nous allons voir ici quel est leur impact sur la frontière de décision.

Tout d'abord, d est le degré du noyau polynomial dont l'expression est donnée à l'Equation 2.15. Lorsque $d = 1$, nous retrouvons le noyau linéaire ; mais à mesure que d augmente, la flexibilité du classificateur augmente, c'est-à-dire sa capacité à discriminer des données non-linéairement séparables. Cet effet est illustré à la Figure 2.11 (toujours en 2-D) : nous pouvons constater dans cet exemple qu'à partir du degré 2 la frontière de décision sépare déjà correctement les données et que le fait de monter jusqu'au degré 5 ne fait qu'accentuer sa courbure.

Cet exemple met en évidence la limite du noyau polynomial, et particulièrement la possibilité que l'augmentation du degré n'améliore plus les performances du classificateur à partir d'une certaine valeur. Autrement dit, il est important de tester graduellement les différents degrés du noyau afin d'éviter qu'il ne soit trop complexe sans gain de performances en retour. Pour notre problème de détection, en utilisant le même algorithme et la même base de données, Lee et al. [45] ont mis en évidence qu'un noyau polynomial de degré 2 était suffisant la plupart de temps ; lorsque ce n'est pas le cas, ils utilisent un degré 4.

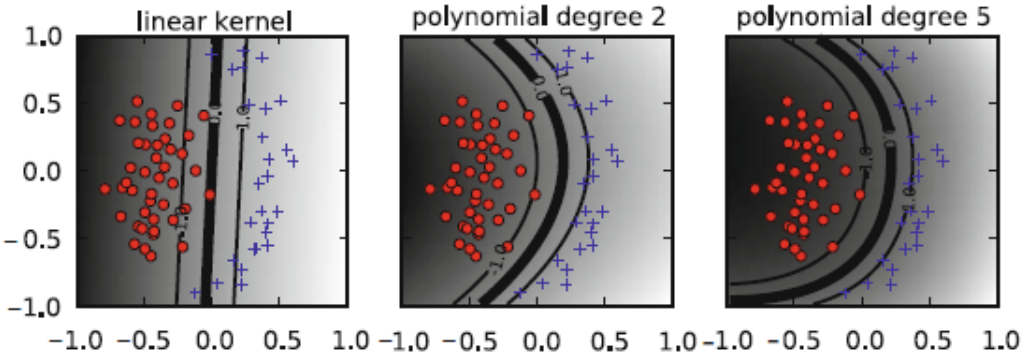


Figure 2.11 – Effet du degré d du noyau polynomial, sur la flexibilité de la fontière de décision [42]

En ce qui concerne le noyau gaussien (Equation 2.16) ; il s'agit d'une fonction de base radiale (RBF) : sa valeur dépend de la distance par rapport à un "centre" - dans notre cas, un vecteur de support $\boldsymbol{SV}_i$ - et, en raison de l'exponentielle négative, décrit une gaussienne selon la norme du vecteur de *features*. La Figure 2.12 illustre cela, pour différentes valeurs du paramètre γ : ce dernier module

donc la largeur de la gaussienne. L'on peut observer qu'au plus γ est faible, au plus la valeur de la fonction noyau évaluée pour un vecteur de *features* $\boldsymbol{X}$ et selon chaque centre $\boldsymbol{SV}_i$ a une chance d'être non-nulle.

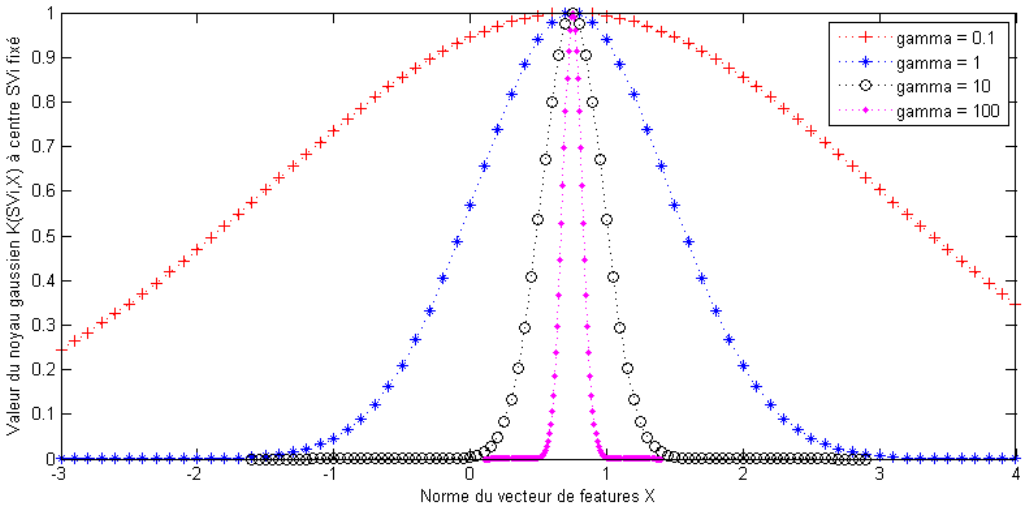


FIGURE 2.12 – Illustration de l'impact du paramètre γ sur la valeur du noyau gaussien, selon la norme du vecteur de *features* $\boldsymbol{X}$, et à centre $\boldsymbol{SV}_i$ fixé (0.75).

Plus concrètement, l'influence du paramètre γ sur la frontière de décision est illustré à la Figure 2.13. Quand γ est très faible, la fonction noyau évaluée pour un vecteur de *features* $\boldsymbol{X}$ a une valeur non-nulle pour l'ensemble des vecteurs de support. Par conséquent, la fonction de discrimination évaluée en $\boldsymbol{X}$ (Equation 2.13) est affectée par le noyau gaussien centré en chacun des vecteurs de support, ce qui résulte en un lissage de la frontière de décision. Par ailleurs, à mesure que γ augmente, le noyau gaussien autour des différents vecteurs de support se rétrécit, impliquant une courbure plus importante de la frontière de décision jusqu'à se limiter aux zones très proches des données. La figure la plus à droite (γ=100) illustre bien le fait que pour des valeurs de γ trop importantes, le classificateur peut se retrouver en *overfitting* ; autrement dit, il "apprend" tellement bien des données d'entraînement qu'il n'est plus capable de classifier correctement les données de test. En ce qui concerne notre problème de détection, Shoeb a également optimisé ce paramètre à une valeur de $\gamma = 0.1$.

Phase d'entraînement : H, K et S. Le détecteur doit être entraîné avec des données issues de $H \geq 24$ heures d'EEG, afin de refléter les différents états physiologiques possibles du patient : éveillé, endormi, les états anormaux non-liés à l'épilepsie, etc. Par ailleurs, Shoeb a montré qu'à partir de $K > 3$ attaques différentes dans le set d'entraînement, le gain en performances est négligeable. Enfin, pour les données d'attaques, le nombre de vecteurs optimal pour l'entraînement correspond à $S = 20$ secondes de crise, à partir du déclenchement électrique, identifié au préalable par un électroencéphalographiste. En effet, à partir de $S > 12$ secondes, le gain en latence devient - assez logiquement - marginal. Par contre, au-delà de $S = 20$ secondes, le nombre d'attaques détectées - et donc la sen-

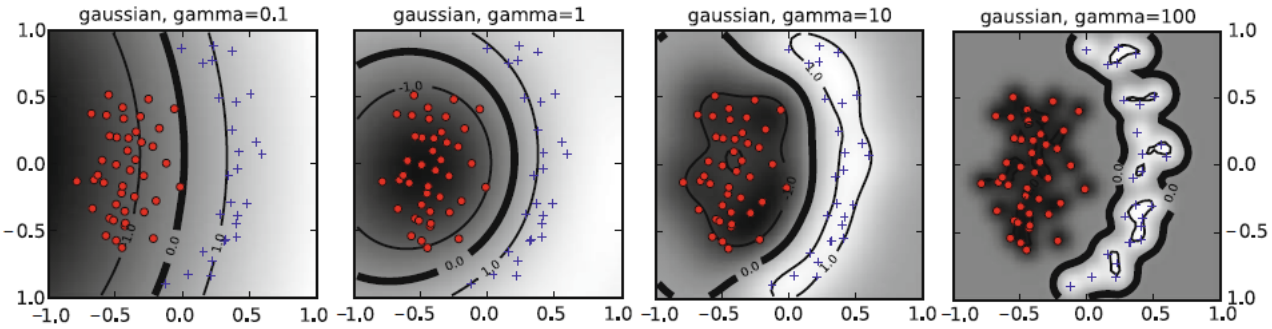


Figure 2.13 – Effet du paramètre γ (inverse de la largeur) du noyau gaussien à facteur de marge souple C fixé : d'une frontière de décision quasi linéaire pour de faibles valeurs, l'on observe un gain de flexibilité avec mesure que γ augmente, jusqu'à mener à un *overfitting* du modèle [42]

sibilité - se stabilise, tandis que le nombre de fausses alarmes augmente, ce qui décroît la spécificité du détecteur. L'explication à cette dernière constatation vient du fait qu'étendre la frontière de décision aux données de "fin" d'attaque accroît la possibilité d'englober des données normales, et donc le nombre de fausses détections.

Chapitre 3

Implémentation software et validation

Dans ce chapitre, nous présenterons une implémentation *software* de l'algorithme décrit au chapitre 2. Celle-ci sera réalisée dans un premier temps sous Matlab, avant un portage en C. Les motivations à cela ainsi que la démarche réalisée seront développées, après une présentation de la base données utilisée ainsi que de la méthodologie d'évaluation des performances du détecteur selon les métriques présentés à la partie 2.2. Enfin, l'algorithme ainsi que les différentes optimisations réalisées à ce stade seront validées et commentées.

3.1 Bases de données

Le développement d'un algorithme ou d'un système de détection de crises d'épilepsie requiert l'utilisation de bases de données pertinentes, qui ne sont pas légions. En effet, il est souhaitable que les données soient issues d'un enregistrement sEEG le plus continu possible. Or, dans l'optique de développer un modèle robuste, nous avons vu à la partie 2.4.3 l'importance de disposer d'au moins 24 heures de données [1], pendant lesquelles le patient doit être sujet à plusieurs attaques (au moins 3). Il est également nécessaire que la base de données recense plusieurs patients - afin de pouvoir réaliser une validation exhaustive de l'algorithme - et soit, pour ce travail, disponible gratuitement. Dans cette partie, nous présenterons dans un premier temps la base de données utilisée. Ensuite, nous verrons pourquoi il aurait été intéressant de disposer d'autres sources, mais également pourquoi cela n'a pas été possible.

3.1.1 Base de données CHB-MIT

Cette base de données est issue de l'enregistrement sEEG de 23 patients pédiatriques [2] souffrant d'épilepsie réfractaire et en évaluation pour une intervention chirurgicale au *Children's Hospital Boston* (CHB). Elle a été conçue en partenariat avec des chercheurs du MIT [3] et est disponible via la banque de données physiologiques gratuites en ligne "Physionet" [18] sous la dénomination "CHB-MIT" [19].

L'EEG des patients a été enregistré pendant plusieurs jours, de façon à obtenir (au moins) 24 heures contiguës contenant un nombre suffisant d'attaques épileptiques. Les différents sets consistent en une série de fichiers au format ".edf" (*European Data Format*, voir partie 3.3.2), contenant 1, 2 ou 4 heures d'enregistrement. Il est à noter toutefois qu'en raison de certaines limitations HW, certains écarts temporels existent entre les différents fichiers : généralement de moins de 10 secondes, mais

1. NB : pas forcément contigües, mais issues de périodes temporelles distinctes afin de construire le modèle en tenant compte des différents états de conscience du patient.
2. 5 garçons et 17 filles, d'âge inférieur à 22 ans
3. Dont Ali H. Shoeb qui a développé l'algorithme que nous implémentons.

parfois plus longs.

Les données sont donc issues d'un EEG non-invasif dont les électrodes ont été placées selon le système international "10-20" et représentent généralement 23 canaux. Les signaux ont été échantillonnés à une fréquence de 256 Hz, avec une résolution de 16 bits. Au total, 182 attaques ont été capturées et labellisées par des experts (électroencéphalographistes), dans plus de 958h d'enregistrements (voir Tableau 3.1).

Patient	Sexe	Age (années)	Durée du set	Nombre d'attaques
Chb01	F	11	40h 55min.	7
Chb02	M	11	35h 26min.	3
Chb03	F	14	38h 00min.	7
Chb04	M	22	156h 06min.	4
Chb05	F	7	39h 00min.	5
Chb06	F	1.5	67h 13min.	10
Chb07	F	14.5	67h 05min.	3
Chb08	M	3.5	20h 00min.	5
Chb09	F	10	68h 27min.	4
Chb10	M	3	50h 02min.	7
Chb11	F	12	34h 19min.	3
Chb12	F	2	24h 09min.	40
Chb13	F	3	33h 00min.	12
Chb14	F	9	26h 00min.	8
Chb15	M	16	40h 00min.	20
Chb16	F	7	19h 00min.	10
Chb17	F	12	21h 00min.	3
Chb18	F	18	36h 03min.	6
Chb19	F	19	30h 32min.	3
Chb20	F	6	28h 02min.	8
Chb21	F	13	33h 23min.	4
Chb22	F	9	31h 00min.	3
Chb23	F	6	21h 29min.	7
Total	5M-18F	9.97 (moy.)	958h 11min.	182

Tableau 3.1 – Caractéristiques des différents patients composant la base de données CHB-MIT

3.1.2 Autres possibilités

Nous avons tenté de trouver d'autres bases de données pertinentes afin de tester noter détecteur, et ce pour deux raisons. Tout d'abord, la base de données CHB-MIT a été conçue avec le concours d'Ali Shoeb qui a développé l'algorithme que nous implémentons ; elle a donc servi à sa validation et il nous paraissait, de ce fait, important de ne pas s'y limiter. Ensuite, elle concerne des sujets pédiatriques, dont l'EEG manifeste une plus grande variabilité entre les états de crise et les états normaux [11] ; à

ce titre, il nous paraissait nécessaire d'utiliser également des données de patients adultes. Cependant, parmi les différentes possibilités rencontrées et exposées ci-dessous, aucune n'est satisfaisante.

La revue de la littérature de Nasehi et al. [9] relève trois bases de données intéressantes, en plus de celle que nous utilisons.

Premièrement, un set conséquent (28 patients, 23 heures/patient, 126 attaques) réalisé par l'Institut et Hôpital Neurologique de Montréal, rattaché à l'Université McGill. Toutefois, celui-ci apparaît introuvable, tant via l'URL référencée [4] que sur le site web de l'hôpital ou de McGill.

Ensuite, plusieurs petits sets sont mis en ligne gratuitement [5] par le *California Institute of Technology* (Caltech) mais dont la longueur n'excède pas les quelques minutes, ce qui est loin d'être suffisant pour nos besoins.

Une autre base de données conséquente (57 patients, 24 heures/patient, 91 attaques) et éprouvée de manière prononcée dans la littérature était fournie [6] par l'Hôpital Universitaire de Freiburg. Cependant, elle fait désormais partie du projet européen "*Epilepsiae*" [7] et, de ce fait, n'est plus disponible gratuitement.

Par ailleurs, nous avons eu connaissance de deux autres bases de données, non-satisfaisantes également en raison de leur taille (très) restreinte : la première [8] provenant de l'Université de Bonn, et la seconde [9] mise en ligne par l'Université de Varsovie.

3.2 Méthodologie d'évaluation des performances

L'évaluation des performances de notre détecteur est réalisée par le biais des métriques présentés à la partie 2.2 : la sensibilité (pourcentage d'attaques détectées), la latence (délai moyen entre le déclenchement électrique d'une attaque et sa détection [10]), la spécificité (nombre moyen de fausses alarmes par heure) et la précision (pourcentage des segments de signaux EEG soumis au détecteur et correctement classés).

La méthodologie de test est un schéma de validation croisée typique en *machine-learning* : le "*leave-one-out*". Le détecteur est entraîné et testé de manière spécifique pour différents patients de la base de données présentée à la partie 3.1. Plus particulièrement, chaque set est composé respectivement de N enregistrements (1 à 4h), dont N_A contenant une ou plusieurs attaques épileptiques, et N_N pour lesquelles le patient est resté dans un état physiologique qualifié de normal. Le "*leave-one-out*" consiste à entraîner N fois le modèle en laissant un des N enregistrements de côté à chaque itération, qui est

4. Base de données SMC : http ://csie.ntu.edu.tw/cjlin/liblinear
5. Base de données KCH : http ://www.vis.caltech.edu/ rodri/data.htm
6. Base de données Freiburg [20] : http ://epilepsy.uni-freiburg.de/freiburg-seizure-prediction-project/eeg-database
7. Projet "*Epilepsiae*" : http ://epilepsy-database.eu
8. Base de données de Bonn : http ://epileptologie-bonn.de/cms/front_content.php ?idcat=193
9. Base de données de Varsovie : http ://eeg.pl/epi
10. Lorsque qu'une attaque n'est pas détectée, elle n'est logiquement pas utilisée dans le calcul de la latence moyenne.

alors le seul utilisé pour la phase de test.

Les calculs de la sensibilité du détecteur et de la latence moyenne sont réalisés sur base des itérations de la validation croisée où l'enregistrement utilisé pour la phase de test est un des N_A contenant une ou plusieurs crises d'épilepsie [8]. Soit D_i une variable binaire indiquant la détection (1) ou non (0) de la i-ème attaque parmi celles présentes dans le set associé à un patient. La sensibilité du détecteur est exprimée par :

$$\text{Sensibilité} = \left(\frac{1}{N_A} \sum_{i=1}^{N_A} D_i \right) \times 100 \qquad \text{avec} \qquad D_i \in 0, 1 \tag{3.1}$$

La latence est déterminée via la moyenne des délais de détection (Latence_i) des R attaques correctement identifiées par le détecteur :

$$\text{Latence} = \frac{1}{R} \sum_{i=1}^{N_A} D_i \times \text{Latence}_i \tag{3.2}$$

Par ailleurs, la spécificité (nombre moyen de fausses alarmes (FA) par heure) est calculée sur l'ensemble de la validation croisée (avec FA_k le nombre de FA pour le k-ème enregistrement du set), et via le nombre T de minutes d'enregistrement utilisées :

$$\text{Spécificité} = \sum_{k=1}^{N} \text{FA}_k \times \frac{60}{T} \tag{3.3}$$

Enfin, la précision du détecteur est exprimée par la moyenne de précision de chaque itération de la validation, déterminant le pourcentage des segments du set de test correctement classés par le détecteur (voir partie 2.2) :

$$\begin{aligned} \text{Précision} &= \frac{1}{N} \sum_{i=k}^{N} \text{Précision}_k \\ \text{avec} \qquad \text{Précision}_k &= \left(\frac{TP + TN}{TP + TN + FP + FN} \right) \times 100 \end{aligned} \tag{3.4}$$

Il est à noter que ce dernier métrique (precision) est donné uniquement à titre indicatif. En effet, dans le cadre d'un détecteur de déclenchement des crises, il n'est pas sujet à optimisation, à l'inverse des trois autres.

3.3 Implémentation Matlab

3.3.1 Motivation

Comme décrit au chapitre 2, nous implémentons dans ce travail un algorithme développé au MIT par Ali H. Shoeb. Cependant, avant d'en réaliser un portage sur notre plate-forme SoC, il nous paraissait judicieux d'en réaliser une première implémentation via Matlab. Les raisons en sont multiples : facilité de manipulation de large matrices de données, fonctions intégrées de traitement de signaux pour extraire les *features* spectraux, facilité de visualisation des données et de debug, etc. En d'autres termes, cela nous paraissait l'outil idéal pour reconstruire l'algorithme de Shoeb ; et ce, malgré le fait que cela requiert une étape d'implémentation supplémentaire par rapport à un portage direct dans le langage C. En outre, c'est par ce support que nous réaliserons une validation exhaustive de notre algorithme.

3.3.2 ToolBoxes spécifiques

Parmi plusieurs possibilités, les *toolboxes* spécifiques présentées ci-dessous ont été choisies d'une part pour leur licence *open-source*, mais surtout en raison de l'existence d'une librairie équivalente dans le langage C, nécessaire pour le portage de notre détecteur sur le microprocesseur ARM Cortex-M0 décrit au chapitre 4.

Bio-Sig. Comme introduit à la partie 3.1.1, les fichiers de la base de données CHB-MIT que nous utilisons sont au format ".edf" (*European Data Format*[11]), l'un des standards de fichiers de données EEG [46]. Afin de pouvoir les traiter facilement sous Matlab, nous avons utilisé la *toolbox* "BioSig". Celle-ci est issue du projet éponyme[12] réalisé à la *Graz University of Technology*, visant à développer une librairie logicielle *open-source*[13] pour le traitement de signaux biomédicaux et sur plusieurs plateformes (Octave, Matlab, C/C++). Les bio-signaux supportés y sont multiples (sEEG, iEEG, ECG, etc.), au même titre que les possibilités de traitement (acquisition, pré-traitement, extraction de *features*, classification, etc.). Notre utilisation s'en est toutefois limitée à la gestion des fichiers ".edf", en raison du fait que cet outil est développé avec l'objectif d'être le plus complet possible et pour être exécuté sur des machines performantes, ce qui n'est, bien entendu, pas notre cas. Pour de plus amples détails, le lecteur est renvoyé vers [47].

SVM-Light. SVM-Light est une implémentation efficiente des SVM en C développée par Joachims Thorsten, à l'Université de Dortmund, puis de Cornell[14] (1998-2008). Ses attraits principaux sont la rapidité de l'algorithme d'optimisation[15] pour la phase d'entraînement des SVM, sa diversité d'ap-

11. Spécifications : http ://www.edfplus.info/specs/edf.html
12. Site web : http ://biosig.sourceforge.net/
13. Licence publique générale GNU
14. Disponible sur la site de Cornell : http ://www.cs.cornell.edu/People/tj/svm_light/
15. Algorithme propre à l'auteur du logiciel ; pour plus de détails, le lecteur est renvoyé vers [48]

plications (classification, régression, classement [16], transduction [17]), ainsi que sa capacité à gérer des modèles SVM complexes (plusieurs milliers de vecteurs de support) ainsi que des larges sets de données d'entraînement.

Outre ces caractéristiques, nous avons choisi d'utiliser cette implémentation des SVM plutôt qu'une autre pour deux raisons. D'une part, la compacité du logiciel - qui aura son importange pour le portage sur le SoC, comme nous le verrons au chapitre 4 - mais surtout la possibilité de réutilisation d'un modèle calculé via Matlab, sur le logiciel en C. En effet, la "*toolbox*" SVM-Light utilisée sur Matlab est en réalité une interface [18] vers le logiciel C originel. Par conséquent, le portage du détecteur en C du classificateur peut se limiter aux fonctions nécessaires pour la phase de test.

3.3.3 Démarche et résultats

Dans sa thèse de doctorat [8], Shoeb décrit de manière assez exhaustive son algorithme ainsi que l'optimisation des différents paramètres relatifs à celui-ci, comme décrit aux parties 2.3.3 et 2.4.3. Par contre, il développe peu les outils utilisés, tant au niveau de l'extraction des *features* (énergie spectrale) que de la classification (SVM) - probablement en raison du fait que la plate-forme cible est peu contrainte, ce qui n'est pas trop cas. Par conséquent, il est à noter que la démarche d'implémentation de l'algorithme réalisée pour ce travail et présentée dans les paragraphes suivants a été guidée par le portage sur la plate-forme SoC cible. En d'autres termes, bien que la première étape d'implémentation soit réalisée via Matlab, les outils utilisés ont été d'emblée limités à ce qui apparaissait comme réaliste pour un système embarqué. De cette façon les résultats obtenus lors de la validation *software* ne sont pas biaisés en regard de l'implémentation *hardware*.

Une fois l'algorithme validé, deux optimisations principales ont été réalisées afin d'optimiser le compromis possible entre performances du détecteur (sensibilité, latence, spécificité) et performances IC du SoC (consommation et taille de puce). Comme il le sera justifié au chapitre 4, la première (décimation des données) a été réalisée dans l'optique d'optimiser la latence de l'extraction des *features* en SW sur le Cortex-M0 et de façon à pouvoir diminuer la fréquence d'opération de la plate-forme, tandis que la seconde (échelonnement des données) a été guidée par l'implémentation de l'accélérateur SVM.

Implémentation alpha (V1). Dans l'optique d'obtenir des résultats probants, la première version du détecteur a été réalisée le plus fidèlement possible aux indications de Shoeb, conformément à ce qui a été décrit aux parties 2.3 et 2.4. En dehors de quelques petites zones d'ombre mineures, les deux inconnues principales de l'algorithme étaient d'une part la façon d'extraire le contenu spectral des

16. En anglais "*ranking SVM*", différencié du problème de classification ("*pattern recognition*").
17. SVM à apprentissage semi-supervisé.
18. Ecrite par Anton Schwaighofer, originellement disponible via http ://www.cis.tugraz.at/igi/aschwaig/index.html (hors service), mais retrouvée via http ://www.codeforge.com/dlpre/150261

données sEEG, et d'autre part, l'algorithme SVM utilisé ; pour ce dernier - et pour les raisons décrites à la partie 3.3.2 - nous utilisons la *toolbox* "SVM-Light".

Il existe une large gamme de possibilités pour dériver le contenu spectral de signaux temporels comme ceux dont nous disposons, spécialement sous Matlab. Toutefois, pour les raisons évoquées plus haut, il ne nous semblait pas pertinent d'utiliser des outils trop sophistiqués. Notre choix s'est donc porté assez classiquement sur l'algorithme FFT, pour deux raisons principales. Tout d'abord, il s'agit d'un algorithme dont le fonctionnement requiert un nombre de points qui soit une puissance de 2. Or, pour rappel, nos données sEEG sont échantillonnées à une fréquence de 256 Hz ; dès lors, nous évitons un "*zero-padding*" de l'algorithme qui bruiterait le spectre et nécessiterait l'utilisation d'une fonction de fenêtrage. Par ailleurs, FFT est désormais un algorithme standard : il a été travaillé et optimisé de manière extensive depuis près de 50 ans. A ce titre, il existe à présent des implémentations convenant à une large gamme de support, dont les systèmes embarqués tels que le nôtre (comme nous le verrons à la partie 4.3).

L'extrait de code 3.1 montre comment les vecteurs de *features* spectraux intermédiaires[19] sont construits à partir de segments de 2 secondes de signaux sEEG (23 canaux), tel que décrit à la partie 2.3.1. Nous pouvons y voir que ce vecteur ($FESpec$) contient, à la sortie, la densités d'énergie spectrale moyenne des 8 sous-bandes de chaque canal. Les opérations sont maintenues les plus simples possibles : par exemple, la FFT est réalisée sans fonction de fenêtrage, et la densité d'énergie spectrale de chaque bande est calculée selon la moyenne des points disponibles et non via une fonction dédiée plus complexe[20].

Extrait de code 3.1 – Construction du vecteur de *features* intermédiaire à partir de 2s de signal

```
1  %(...)
2  Spec = fft(signal,512)/512;              % FFT (normalisee) sur 512 points pour
3                                           %    2s de signal (512 echant. x 23 canaux)
4  BPI = 2*abs(Spec(1:(24*2),:));           % Tronque selon la bande d'interet (24Hz)
5                                           %    et conversion en "single-sided spectrum"
6  ESD = 20*log10(BPI);                     % Densite d'energie spectrale [dB]
7  ESDresh = (reshape(ESD,6,[]))';          % Remodelage en ((23x8) x 6), soit les 8
8                                           %    sous-bandes de chaque canal a la suite
9  for j=1:FVSize                           % Le vecteur final contient la moyenne de
10     FESpec(j) = sum(ESDresh(j,:))/6;     %    chaque sous-bande (184x1), afin de dimi-
11 end                                      %    minuer le nombre de features.
12 %(...)
```

19. Pour rappel, le vecteur de *features* complet est construit en concaténant W=3 vecteurs intermédiaires caractérisant trois segments temporels de 2 secondes contigus, mais ne se chevauchant pas.
20. Par exemple "bandpower".

Le Tableau 3.2 reprend les performances de détection calculées selon la méthodologie décrite à la partie 3.2, pour plusieurs patients de la base de données CHB-MIT et en comparant différentes versions de l'algorithme. L'idée est de mesurer l'impact des optimisations - présentées par la suite - réalisées dans l'optique des performances IC du SoC cible, sur les performances du détecteur. Afin de ne pas le surcharger, seules les données du noyau SVM le plus performant sont reprises. Pour chaque patient, la première ligne reprend les résultats de Lee et al [45] ; la comparaison avec celui-ci était la plus pertinente, étant donné qu'il utilise également le base de données CHB-MIT et l'algorithme de Shoeb mais, à l'inverse de ce dernier, il ne se cantonne pas au noyau RBF pour dériver le modèle SVM et teste, comme nous, le linéaire, les polynomiaux de degré 2 et 4, et le RBF.

Patient	Algorithme [a]	Noyau SVM	Précision [%]	Sensibilité [%]	Latence [s]	Spécificité [FA/h]
Chb01	Lee et al [45]	Linéaire		100	4.29	0.00
	V1	Linéaire	99.78	100	2.86	0.54
	V2	Linéaire	99.89	100	3.86	0.00
	V3	Linéaire	99.87	100	3.71	0.02
	V3+SSC [b]	Linéaire	99.84	100	3.71	0.06
Chb02	Lee et al [45]	Poly 2	-	100	5.67	0.06
	V1	Linéaire	99.77	100	4.33	0.23
	V2	Poly 4	99.89	100	5	0.23
	V3	Poly 2	99.85	100	6	0.08
Chb03	Lee et al [45]	Poly 4	-	100	1.86	0.03
	V1	Poly 2	99.83	100	2.29	0.68
	V2	Poly 4	99.86	100	2.86	0.32
	V3	Poly 4	99.92	100	3	0.29
Chb05	Lee et al [45]	Poly 2	-	100	4.2	0.1
	V1	Poly 2	99.88	100	4	0.23
	V2	Poly 2	99.96	100	4.8	0.08
	V3	Poly 2	99.96	100	5	0.08
Chb08	Lee et al [45]	Linéaire	-	100	3.8	0.03
	V1	Linéaire	98.69	100	4	1.50
	V2	Poly 2	98.47	100	6.4	0.15
	V3	Poly 2	98.45	100	6.8	0.10
Chb10	Lee et al [45]	Poly 4	-	100	4.29	0.02
	V1	Poly 2	99.91	85.7	3.17	0.56
	V2	Poly 2	99.90	100	3.15	0.26
	V3	Poly 2	99.96	100	3.71	0.12
Moyenne	Lee et al [45]	-	-	100	4.02	0.04
	V1	-	99.64	99.25	3.44	0.49
	V2	-	99.66	100	4.35	0.17
	V3	-	99.67	100	4.70	0.12

a. V1 = Alpha / V2 = Décimation / V3 = Échelonnement
b. Utilisation d'un sous-set de canaux déterminé par un algorithme de "*greedy search*"

Tableau 3.2 – Comparaison pour plusieurs patients du set CHB-MIT des performances de différentes versions de l'algorithme de détection implémenté via Matlab

Les résultats du Tableau 3.2 montrent que notre détecteur atteint une sensibilité de près de 100%. Il est donc plus intéressant de s'attarder aux variations de latence (Fig. 3.1) et de spécificité (Fig. 3.2), métriques que nous cherchons tous deux à minimiser. Nous pouvons observer que cette première version de notre algorithme affiche une meilleure latence moyenne que celle de Lee, mais présente, *per contra*, une spécificité nettement supérieure. Il est probable que cette différence vienne l'outil d'extraction des *features* spectraux utilisé : Lee réalise cela par le biais d'un accélérateur *hardware* et utilise, de ce fait, une banque de filtres passe-bande. Nous verrons toutefois que ces différences vont se restreindre pour les deux autres versions de l'algorithme.

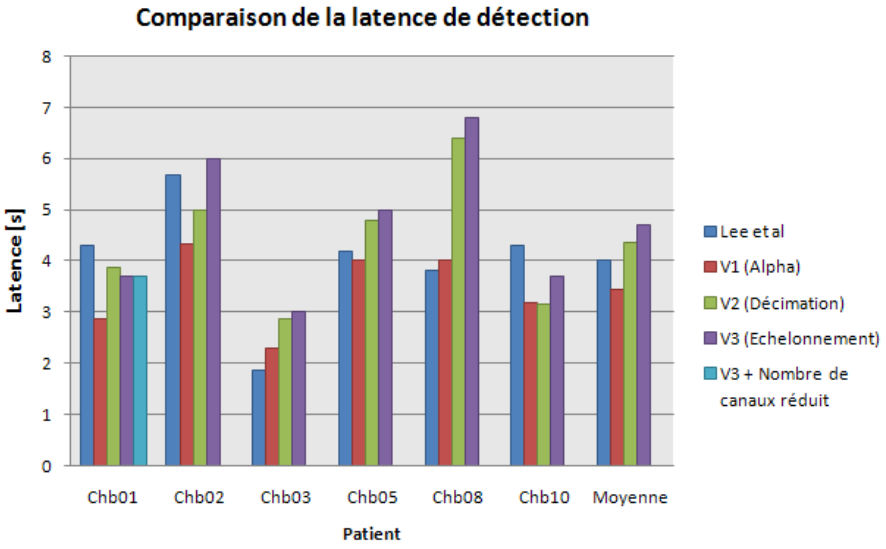


Figure 3.1 – Comparaison de la latence de détection pour les différentes versions de l'algorithme implémentées

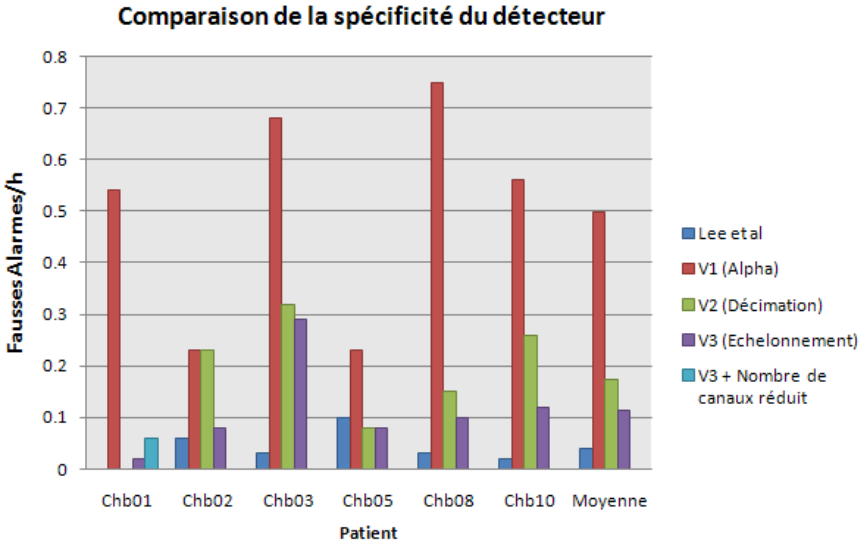


Figure 3.2 – Comparaison de la spécificité du détecteur pour les différentes versions de l'algorithme implémentées

Décimation des données (V2). D'un point de vue calculatoire, l'extrait de code 3.1 reprend la partie la plus coûteuse de l'algorithme. A présent que celui-ci est validé comme étant performant sous cette forme, il convient de tenter de l'optimiser, dans l'optique du portage SW sur le SoC. A cet effet, la V2 part du constat suivant : étant donné que la bande d'intérêt - limitée à 24 Hz - est nettement moins importante que la fréquence d'échantillonnage (256 Hz), la transformée de Fourier peut être réalisée sur moins de points. Etant donné que celui-ci doit être une puissance de 2, nous choisissons de n'utiliser que 32 points par seconde (donc 64 pour un segment de 2 secondes), ce qui implique de décimer les données du signal d'un facteur 8.

De cette façon, le coût calculatoire de l'algorithme est réduit de ce même facteur 8. Or, le Tableau 3.2 ainsi que les Figures 3.1 et 3.2 montrent que l'impact sur les performances du détecteur est acceptable : en effet, s'il est vrai que la latence moyenne augmente 51 secondes, le nombre de fausses alarmes est réduit de près d'un facteur 3, rapprochant nos performances de celle de Lee. Et comme nous le verrons à la partie 4.3, cela nous permettra de diminuer de façon importante la fréquence d'opération du SoC.

Échelonnement des données (V3). La deuxième amélioration est inspirée du fait que les SVM répondent mieux aux données normalisées [42]. Cette normalisation peut être réalisée de diverses manières, et ce, soit au niveau des *features* en tant que tels (p. ex. en divisant chaque vecteur de *features* par sa norme euclidienne), soit au niveau du noyau : en effet, nous avons vu à la partie 2.4.1 que la méthode des noyaux permet de calculer un produit scalaire entre deux vecteurs sans devoir projeter explicitement les données dans l'espace des caractéristiques. Par conséquent, il en va de même pour la norme d'un vecteur, en redéfinissant le noyau de la façon suivante :

$$\bar{K}(\boldsymbol{X}, \boldsymbol{Z}) = \frac{K(\boldsymbol{X}, \boldsymbol{Z})}{\sqrt{K(\boldsymbol{X}, \boldsymbol{X}) K(\boldsymbol{Z}, \boldsymbol{Z})}} \tag{3.5}$$

Toutefois, la normalisation des données augmente la charge calculatoire, contrairement à ce que nous cherchons à réaliser. En outre, la nécessité de normaliser les données de la sorte est généralement pertinente lorsque les différents *features* proviennent de données qui n'obéissent pas à la même échelle [21], ce qui n'est pas notre cas. Par contre, cela nous apprend de manière intuitive que la SVM est capable de classer correctement des données qui sont échelonnées (multipliées ou divisées) de la même façon.

Ce constat est utilisé pour optimiser notre algorithme à deux égards. Tout d'abord, pour alléger au maximum l'extraction des *features* spectraux (voir extrait de code 3.2), en se débarrassant des opérations qui ne réalisent qu'une factorisation des données. Ensuite, nous choisissons d'effectuer une pseudo-normalisation des données en divisant chaque *feature* par un même facteur : le *features* de valeur la plus élevée dans la matrice d'entraînement complète. Le bénéfice de cette action est d'avoir

21. Il est alors important de "standardiser" les données en soustrayant à chaque *feature* la moyenne du vecteur et en divisant cette différence par l'écart-type

tous nos *features* compris entre 0 et 1 ; l'intérêt à cela réside dans la possibilité de limiter alors la précision utilisée pour représenter ces données. Et comme nous verrons à la partie 4.4, cela nous servira à diminuer d'un facteur 2 la consommation de l'accélérateur SVM. Or, du point de vue des performances de détection (Fig. 3.1 et 3.2), nous voyons que la latence a encore un peu augmenté, mais que la spécificité est à nouveau réduite.

Extrait de code 3.2 – Version optimisée de la construction du vecteur de *features* intermédiaire

```
1  %(...)
2  Spec = fft(signal,64);                % FFT sur 64 points pour 2s de signal
3                                        % (64 echant. x 23 canaux)
4  BPI  = abs(Spec(1:(24*2),:));         % Tronque selon la bande d'interet (24Hz)
5                                        % et conversion en "single-sided spectrum"
6
7  BPIresh = (reshape(BPI,6,[]))';       % Remodelage en ((23x8) x 6), soit les 8
8                                        % sous-bandes de chaque canal a la suite
9
10 for j=1:FVSize                        % Le vecteur final contient la moyenne de
11     FESpec(j) = sum(BPIresh(j,:))/6;  % chaque sous-bande (184x1), afin de dimi-
12 end                                  % minuer le nombre de features.
13 %(...)
```

Complexité du noyau SVM. En marge de la complexité calculatoire de l'extraction des *features* spectraux - réduite par le biais des optimisations présentées ci-dessus - d'autres facteurs influencent le coût énergétique de notre algorithme de détection. L'un de ces facteurs est la complexité du noyau SVM utilisé. Or, le Tableau 3.2 montre une tendance de meilleures performances pour les noyaux de plus faible complexité. Bien que nous n'ayons pas pu validé notre algorithme sur l'ensemble de la base de données CHB-MIT, cette tendance est confirmée de façon exhaustive [22] par [45]. C'est pourquoi nous limiterons le développement de notre accélérateur HW pour la phase de classification de l'algorithme SVM aux noyaux linéaire et polynomial de degré 2, toujours dans l'objectif de basse-consommation.

Réduction du nombre de canaux sEEG. Les autres facteurs influençant le coût énergétique d'une détection sont les paramètres algorithmiques de construction du vecteur de *features* discutés à la partie 2.3.3 [23] : la durée des segments de signaux sEEG utilisés (L), le nombre de canaux sEEG

22. Sur 22 patients de la base de données CHB-MIT, le noyau linéaire donne les meilleures performances pour 5 d'entre eux, 10 s'en tiennent au noyau polynomial de degré 2 et 2 à celui de degré 4, tandis que 5 seulement nécessitent un noyau RBF.

23. Par contre les paramètres liés à la classification et discutés à la partie 2.4.3 n'impactent que la phase d'entraînement et leur optimisation ne vise qu'à améliorer les performances de détection

utilisés (N), le nombre de sous-bandes par lequel le spectre des signaux est divisé (M) et le nombre de vecteurs intermédiaires utilisés pour former le vecteur de *features* complet (W). En effet, leur valeur va déterminer le nombre de *features* à extraire, soit le nombre de fois que la FFT devra être utilisée ainsi que la mémoire de données nécessaire.

Les paramètres L, M et W ont été sujet à optimisation de la part de Shoeb. Par conséquent nous en garderons les valeurs fixées au chapitre 2. A l'inverse, nous avons vu qu'il utilise par défaut le nombre maximum de canaux sEEG disponibles [24]. Or nous avons également mentionné que Shih et al [38] avaient obtenus des résultats prometteurs en appliquant un algorithme de sélection des *features* aux canaux sEEG : pour une base de données différente de la nôtre, cela leur a permis de diminuer de près 70% la consommation d'un système ambulatoire sur lequel ils avaient réalisés un portage SW d'un algorithme de détection (différent du nôtre également), et ce, au prix de pertes acceptables dans les performances de détection [25]. En outre, le même genre de résultats a été obtenu par Lee et al [45], pour la base de données CHB-MIT, mais pour quatre patients uniquement.

Afin d'éviter de devoir réaliser une recherche exhaustive [26] du meilleur sous-set de canaux à utiliser, une alternative consiste à utiliser algorithme de "*greedy search*". Le meilleur sous-set y est déterminé de façon incrémentale via N itérations ; deux approches sont possibles, la sélection "directe" ou la sélection "inverse". Dans le premier cas (utilisé par [45]), chaque itération implique l'ajout au sous-set du canal donnant le meilleur gain en performances, tandis que dans le second (utilisé par [38]), nous démarrons avec l'ensemble des canaux, et chaque itération consiste en l'élimination du canal dont la mise à l'écart induit le meilleur [27] résultat.

Etant donné qu'elle a été validée de façon plus importante et parce que nous n'avions pas la possibilité de réaliser les deux dans ce travail, nous choisissons d'utiliser la deuxième méthode. En effet, s'il vrai qu'elles sont moins coûteuses qu'une recherche exhaustive, elles nécessitent tout de même réaliser N fois la validation croisée décrite à la partie 3.2 pour un même patient. Pour cette raison, nous n'avons pu la réaliser qu'à titre indicatif, pour le patient "Chb01" (voir Tableau 3.2). Toutefois les résultats sont très prometteurs puisque nous avons pu identifier un sous-set de trois canaux (en utilisant la V3 de l'algorithme) pour lesquels seule la spécificité est dégradée d'un facteur 3. Nous verrons au chapitre 4 que ce résultat (et *a fortiori* ceux de [38] et [45]) influera sur la conception de l'accélérateur *hardware* SVM.

24. Pour une raison inconnue, Shoeb ne disposait que de 18 canaux pour la même base de données. A aucun moment il ne cite l'existence de 23 canaux. Or, dans la sélection des canaux réalisés, ceux dont nous disposions en plus se sont révélés déterminants.

25. Pour rappel, 2% de sensibilité en moins, 2s de latence supplémentaire, mais une diminution du taux moyen de fausses alarmes d'un facteur 2.

26. Qui nécessiterait, pour N canaux, de réaliser 2^N-1 fois la validation croisée décrite à la partie 3.2

27. Ou le moins mauvais, c'est selon.

Chapitre 4

Plateforme "system-on-chip"

Dans ce chapitre, nous présenterons la plate-forme de type "*system-on-chip*" (SoC) développée dans le cadre de ce travail. Dans un premier temps, nous en donnerons une vue d'ensemble, en présentant les différents éléments qui la composent, ainsi que leur fonction dans la détection des crises d'épilepsie. Ensuite, nous présenterons plus en détails le microprocesseur ARM Cortex-M0 équipant notre plate-forme, ainsi que le portage logiciel en C de l'algorithme de détection. Enfin nous verrons comment la partie de classification a pu être implémentée via un accélérateur HW dédié, ainsi que son apport tant au niveau des performances du détecteur que de l'objectif de basse consommation.

4.1 Vue d'ensemble

Pour rappel (voir partie 1.3.3), les signaux sEEG sont extraites par le biais de plusieurs électrodes placées sur le scalp du patient. Les différents canaux sont formés en prenant la différence de potentiel entre deux électrodes, qui est alors traitée par un module analogique de conditionnement (amplification, filtrage). Ensuite les données sont échantillonnées par un ADC : dans notre cas, à 256 Hz et 16 bits de résolution. Enfin, comme précisé au chapitre 3, les données sont décimées par un facteur 8 via un module hors-puce [1] qui réalise également une conversion parallèle-série et fait dès lors office de tampon entre les parties analogiques et digitales du système global.

C'est à ce niveau que notre plate-forme intervient (Fig. 4.1). Le contrôle des opérations est assuré par un module dédié nommé ESOD [2]. Lorsque des données sEEG sont disponibles, l'unité de décimation fait une requête d'écriture en mémoire des signaux (SRAM "On-chip" 3) au module ESOD. Cette mémoire est de 8kB [3], afin de pouvoir contenir deux fois le nombre de données caractérisant un segment sEEG de 2 secondes et lorsque le nombre de canaux utilisés est le plus élevé (23), soit 5,9kB [4]. Etant donné que nous ne disposons que de mémoires mono-port, le module ESOD gère les conflits d'accès aux données sEEG entre le décimateur (écriture) et le Cortex-M0 (lecture, a la priorité). Il est à noter que la taille de mémoire est doublée par rapport à la place nécessaire pour un segment de données en raison du fait que le Cortex met moins de temps a lire et traiter les données que n'en prend l'écriture (au moins 2 secondes, bien entendu).

La canal de communication principal du SoC est une interface AHB-Lite [54]. C'est par ce biais que le Cortex-M0 est capable d'accéder aux différents périphériques, qui peuvent, de leur côté, lui adresser

1. NB : le décimateur est considéré hors puce car n'a pas été caractérisé dans ce travail, étant donné que les données étaient placées dans la mémoire à l'initialisation ; toutefois, dans un développement ultérieur, il se trouverait sur la puce.
2. "*Epileptic Seizure Onset Detector*"
3. En raison des tailles de mémoire disponibles ; autrement, une mémoire plus petite aurait convenu.
4. Par segment de deux secondes, 64x23x2 = 2,94 kB

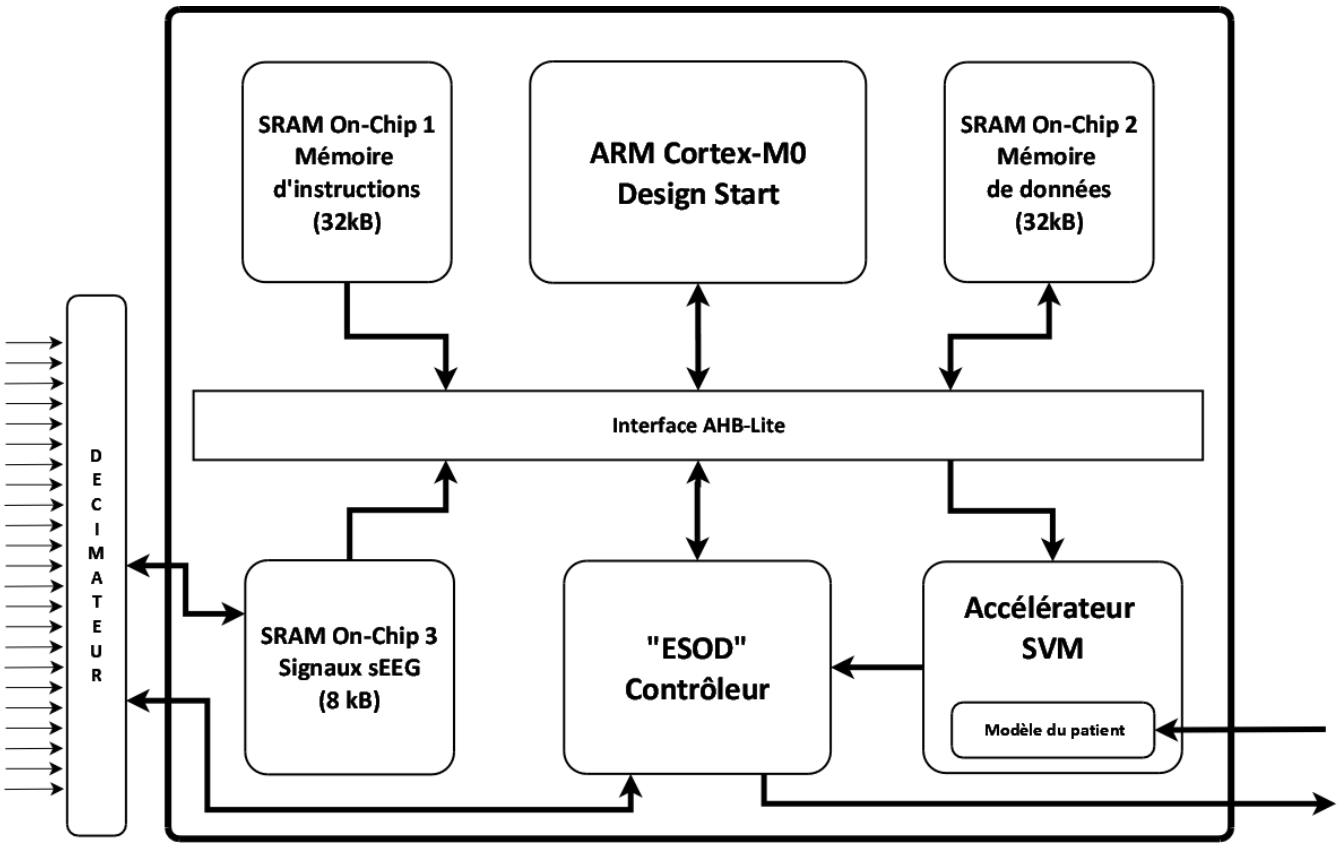


Figure 4.1 – Schéma-bloc de la plate-forme SoC développée

des interruptions. Outre la mémoire des signaux déjà présentée, trois autres SRAM se trouvent sur la puce : les mémoires d'instructions et de données du Cortex (32 kB chacune), ainsi qu'une plus petite (4kB) servant à stocker le modèle de la SVM, ainsi que les données temporaires nécessaires à l'accélérateur.

Par ailleurs, seul l'accélérateur SVM a la possibilité de contourner le bus AHB-lite, afin de transmettre au module ESOD le résultat de détection de chaque segment de deux secondes. De son coté, le module ESOD est le seul à pouvoir communiquer avec l'extérieur, en dehors de l'initialisation du modèle SVM spécifique au patient, ainsi que de l'écriture des données sEEG dans la SRAM associée.

Etant donné que l'algorithme doit dériver le contenu des signaux sEEG sur l'ensemble d'un segment de deux secondes, la plate-forme ne peut anticiper les opérations, ce qui conditionne la latence globable du système de détection. Comme nous le verrons à la partie 4.2, le Cortex-M0 est par défaut en état de "sommeil", dans lequel sa consommation est minmisée. Lorsqu'un segment sEEG complet est disponible, le module ESOD le "réveille" par le biais d'une interruption. L'extraction des *features* est réalisé de manière logicielle sur le Cortex, tandis que la classification peut être effectuée sur le Cortex, ou par le biais d'un accélérateur SVM *hardware* dédicacé.

Lorsque le détecteur est entièrement logiciel, l'accélérateur est désactivé ("*clock-gating*"). Dans le second cas, le Cortex traite les données canal (sEEG) par canal et envoie le vecteur de *features*

intermédiaires caractérisant le segment sEEG courant à l'accélérateur. Comme nous le verrons par la suite, dans le cas du noyau linéaire l'accélérateur peut anticiper les calculs en utilisant les vecteurs intermédiaires caractérisant les segments précédents et qu'il stocke dans sa mémoire interne, mais pas dans le cas du noyau polynomial. Enfin, le résultat de détection est transmis vers l'extérieur par le module ESOD.

4.2 Microprocesseur ARM Cortex-M0

Le microprocesseur utilisé dans notre SoC est un ARM Cortex-M0 [52]. Il s'agit d'un processeur 32-bit de type RISC [5] avec une architecture "von Neumann" (mono-bus). Conçu pour être intégré dans des systèmes à ultra-basse-consommation, il est à la fois dense et efficace d'un point de vue énergétique.

En effet, sa synthèse requiert (au minimum) 12000 portes logiques [49] - soit autant qu'un microprocesseur 16-bit typique - tout en réalisant 0,9 DMIPS/MHz selon le benchmark Dhrystone (soit nettement plus que les microprocesseur 16-bit typiques, voir Fig. 4.2), et avec une consommation minimum de 12 μW/Mhz [6]. A fortiori, du fait d'être bien plus rapide, le Cortex-M0 peut rester en moyenne plus de temps à l'état "sommeil" (faible-consommation) - comme développé plus bas - entre deux interruptions ; par conséquent son utilisation permet de réduire la consommation moyenne, sans perte de performances par rapport aux solutions 16-bit.

Par ailleurs les solutions 32-bit typiques, bien que plus performantes que le Cortex-M0, ont une taille, et donc une consommation moyenne, plus importantes que le Cortex. Dès lors, il affiche un compromis intéressant entre taille, performances et consommation, ce qui en fait le choix idéal pour notre plate-forme.

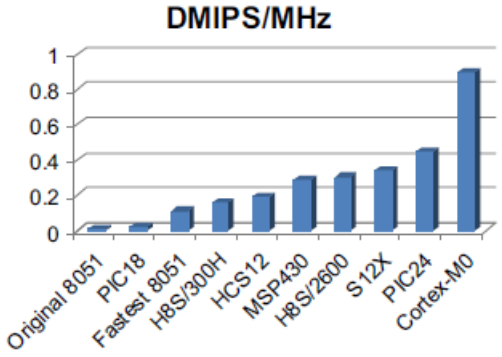


Figure 4.2 – Comparaison Dhrystone de plusieurs microprocesseurs faible-consommation [49]

Origine du Cortex-M0. La famille des microprocesseurs Cortex de ARM Ltd est destinée aux systèmes embarqués. Il en existe trois catégories qui se différencient par l'application cible (voir Fig.

5. "*Reduced Instruction Set Computing*"
6. Avec une technologie CMOS 65nm (comme dans notre cas) ; pour une granularité à 180nm, cela monte à 85μW/Mhz.

4.3). De cette façon, les Cortex-A sont des processeurs dits "d'application" (p.ex. smartphones, tablettes, etc.), les Cortex-R conviennent bien aux systèmes temps-réel à haute performance tandis que les Cortex-M sont destinés généralement aux unités microcontrôleur (MCU), plus exigeantes en termes de consommation.

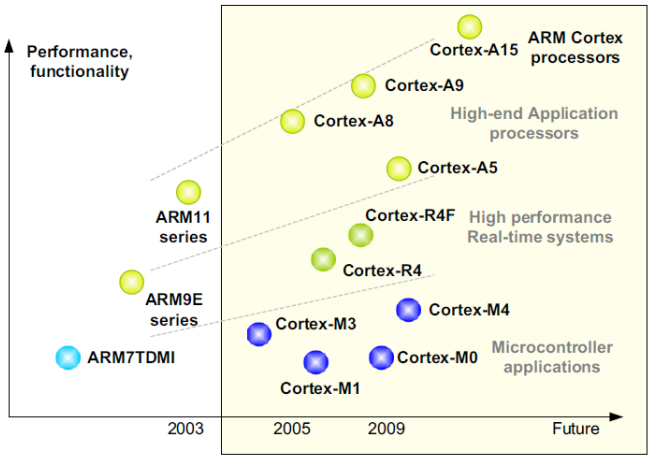


FIGURE 4.3 – La famille des microprocesseurs ARM Cortex divisée en trois catégories selon l'architecture d'origine et l'application cible [49]

Pour chaque génération de processeurs, ARM spécifie une nouvelle architecture capable de gérer de nouveaux *features*[7]. Or la version ARMv7 a vu apparaître les différents sous-sets correspondant aux catégories mentionnées de Cortex, afin satisfaire à leurs contraintes spécifiques. A ce titre, les Cortex-A9/R4/M3 supportent respectivement l'architecture ARMv7-A/R/M.

Fort du succès de son Cortex-M3, ARM a décidé de s'étendre dans le domaine des microcontrôleurs. Dans cette optique, ils ont effectué une refonte de l'architecture ARMv6 en récupérant des *features* de l'ARMv7-M pour produire l'ARMv6-M [53], dont sont issus les Cortex-M0 et M1 : le premier destiné aux applications à ultra-basse-consommation, et le second pour les implémentations sur FPGA.

Cortex-M0 DesignStart L'architecture ARMv6-M est concise, efficiente, et dotée de nombreux *features*. Parmi ceux-ci, deux nous intéresserons particulièrement dans ce travail : d'une part, les mécanismes efficaces d'interruption (via un contrôleur NVIC[8], voir Fig. 4.4), ainsi que les *features* de faible consommation disponibles tels que les différents niveaux de sommeil.

En effet, il est à noter que nous disposons d'une version "DesignStart" [50, 51] du Cortex-M0, limitées au niveau des *features* disponibles (voir Fig. 4.5). Outre le multiplicateur en 1 cycle d'horloge (contre 32) et la taille minimum possible de 12000 portes logiques, il aurait été intéressant de disposer des mécanismes de faible-consommation avancés, tels que l'unité WIC (voir Fig. 4.4). Celle-ci permet

7. Par exemple l'architecture ARMv4T a apporté la possibilité d'un debugging JTAG ainsi qu'un multiplicateur rapide.
8. "*Nested Vectored Interrupt Controller*"

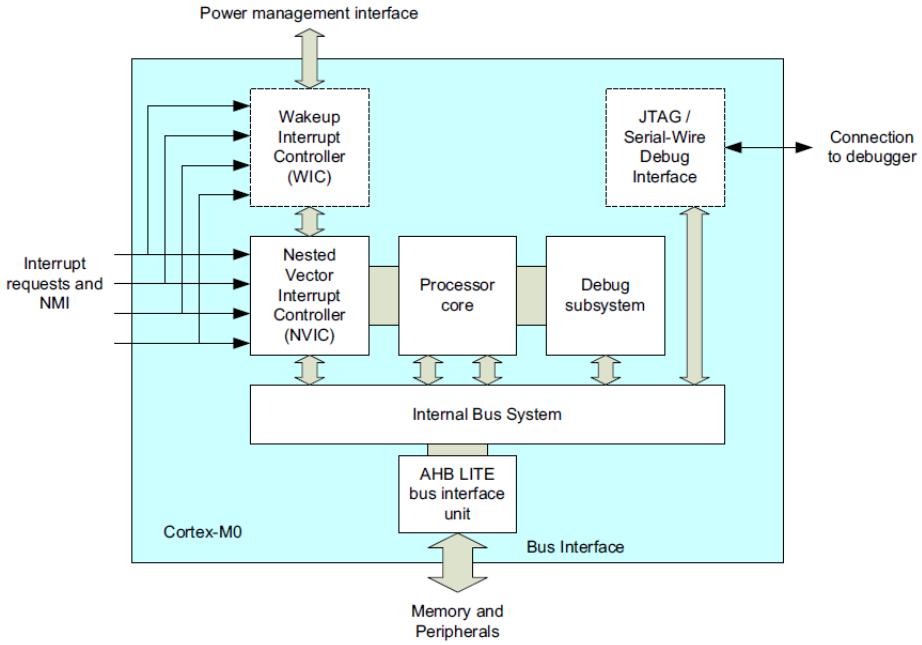


Figure 4.4 – Schéma-bloc simplifié du Cortex-M0 complet [49]

un état de *standby* profond où le processeur ainsi que le NVIC sont entièrement coupés ; or, le WIC permet au processeur de rester sensible aux interruptions et, le cas échéant, réalise un "*power-up*" efficace du Cortex.

Options de faible-consommation Les besoins de notre application ne requièrent pas l'ajout d'un système d'exploitation temps-réel. En l'absence de celui-ci, une façon de programmer le Cortex est d'utiliser une boucle infinie ("*polling*", voir Fig. 4.6a) qui, à chaque itération relèverait l'état des différents périphériques de manière séquentielle. Les deux désavantages d'un tel schéma sont d'une part, le manque de flexibilité et d'autre part le fait que le processeur est toujours en marche, même lorsque l'algorithme ne le nécessite pas. Ce deuxième constat est le plus problématique dans notre cas.

C'est pourquoi nous adopterons plutôt le schéma illustré à la Figure 4.6b qui consiste à mettre le Cortex dans un état de "sommeil" lorsque aucune action de sa part n'est requise et de le réveiller par le biais d'une interruption lorsque nécessaire. En outre, cette façon de procéder est facilitée par les différents *features* du Cortex. Tout d'abord, le bus AHB-Lite [54] permet de véhiculer facilement les interruptions, qui sont gérées efficacement et de manière interne au Cortex par une unité NVIC dédiée (voir Fig. 4.4). De plus, les librairies CMSIS [9] permettent une configuration aisée des différentes interruptions : activation/désactivation, priorisation, association aux différents périphériques, etc. Par ailleurs, l'état de sommeil du Cortex est configurable selon deux niveaux ("normal" et "profond", toujours via le CMSIS), et lorsque la gestion d'une interruption est terminée, l'option "*sleep-on-exit*" permet de le remettre efficacement dans l'état de sommeil.

9. "*Cortex Microcontroller Software Interface Standard*"

ARM Cortex-M0 processor features	Full product options	"M0_DS" implementation
Verilog core	✓	Flattened and Obfuscated
AMBA AHB-lite interface	✓	✓
ARMv6-M instruction set architecture	✓	✓
NVIC Interrupt controller	✓	✓
Interrupt line configurations	1 to 32	16 only
Debug (SWD, JTAG) option	✓	
Up to 4 breakpoints, 2 watchpoints	✓	
Low power optimisations (ACG)	✓	
Multiple power domain support with WIC	✓	
Fast multiplier (1 cycle) option	✓	
System timer	✓	✓
Area (gates)	**12k – 25k**	**16K**

FIGURE 4.5 – Limitations du Cortex-M0 DesignStart par rapport au Cortex-M0 complet [51]

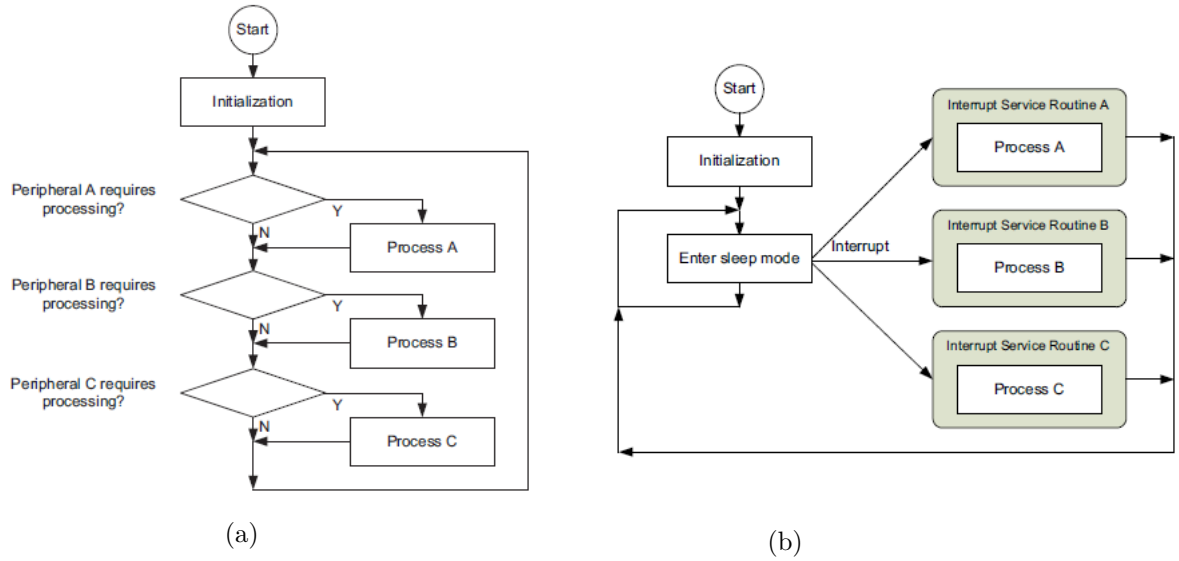


FIGURE 4.6 – Schémas de programmation temps-réelle possibles pour le Cortex-M0 en l'absence d'un système d'exploitation : (a) polling (b) interruptions [49].

4.3 Détecteur software

La première implémentation du détecteur sur le SoC a consisté en un portage entièrement logiciel de l'algorithme. Pour ce faire le code matlab utilisé pour valider l'algorithme a été transcrit en C. Aucune optimisation en assembleur n'a été réalisée étant donné que l'architecture des Cortex-M est déjà optimisée pour supporter les programmes en C [49]. Dans cette partie, nous décrirons dans un premier temps ce portage, puis nous validerons les différentes optimisations *software* introduites au chapitre 3. Les performances IC du système lorsque la détection est réalisée de manière entièrement logicielle seront discutées à la partie 4.5.

4.3.1 Portage en C

D'un de point de vue *software*, la contrainte majeure du portage de l'algorithme sur notre SoC est la taille du code, qui va, de manière évidente, déterminer la taille de mémoire d'instructions nécessaire. Pour les fonctions spécifiques, il a donc été important d'utiliser des outils réalisant le meilleur compromis possible entre compacité et rapidité d'exécution.

Nous avons vu aux chapitres précédents que l'algorithme implémenté consiste en deux phases principales : l'extraction de *features*, dont la complexité est gouvernée par la dérivation du contenu spectral des signaux sEEG d'entrée, et la classification sur base d'un modèle SVM. Comme précisé au chapitre 3, le calcul de ce modèle est assez coûteux, mais peut être réalisé au préalable et en dehors du SoC. En outre, nous avons vu que l'utilisation de la librairie SVM-Light nous permet de réutiliser les modèles obtenus via Matlab. Par conséquent, le portage de la partie classification a, dans un premier temps, consisté en l'intégration dans notre programme des fonctions de classification du programme SVM-Light. Toutefois, comme nous le verrons à la partie 4.4, nous utiliserons une reformulation des noyaux linéaires et polynomial de degré 2. Dès lors, afin de supporter les mêmes modèles en SW et en HW, nous avons réécrit ces fonctions.

En ce qui concerne l'extraction des *features* spectraux, nous avons validé au chapitre 4 l'utilisation de l'algorithme FFT plutôt que d'autres outils d'analyse fréquentielle plus avancés. De plus - et comme mentionné précédemment - sa popularité depuis de nombreuses années implique l'existence d'une large variété d'implémentations possibles. Parmi celles-ci, la librairie "Kiss [10] FFT" [11] se démarque pour nos besoins : en effet, en plus d'être sous license gratuite [12], elle est raisonnablement efficiente pour une taille de code restreinte (entre 10 et 20 KB selon les options utilisées).

10. "*Keep It Simple Stupid*", principe de développement selon lequel la simplicité est considérée comme un objectif de design clé, et prônant par conséquent la nécessité d'éviter toute complexité inutile.
11. Site web : http ://sourceforge.net/projects/kissfft/.
12. License BSD : http ://www.opensource.org/licenses/bsd-license.php

4.3.2 Validation des optimisations

Lors de la validation de l'algorithme au chapitre 3, nous avons réalisé deux optimisations principales : dans un premier temps, nous avons choisi de décimer les données afin de réaliser la FFT sur 64 points plutôt que 512. Puis, nous avons échelonné les données via une pseudo-normalisation, ce qui nous a permis d'alléger l'extraction du contenu spectral des signaux sEEG. D'un point de vue *software*, le but de ces optimisations est de diminuer la latence de la phase d'extraction des *features* (FE), afin de pouvoir réduire la fréquence d'opération du système, et donc la consommation globale.

Le Tableau 4.1 reprend pour chaque version du *software* la latence de la phase de FE, à 10Mhz, et pour un segment de deux secondes contenant 23 signaux sEEG. La dernière colonne projette la fréquence minimum de fonctionnement possible estimée sur base de résultats à 10 Mhz et gouvernée par la contrainte temps-réel de traitement successif des segments de deux secondes. Toutefois, comme nous le verrons par la suite, ces fréquences sont légèrement optimistes étant donné que, même lorsque la classification est réalisée via l'accélérateur HW, d'autres opérations sont nécessaires (p.ex. la communication entre le Cortex et l'accélérateur).

Les différentes optimisations ont bien l'effet escompté : comme attendu, la décimation des données apporte le meilleur gain en latence. Ensuite, la V3 diminue encore un peu la latence en se débarrassant des opérations superflues (factorisations) dans la dérivation du contenu spectral. La dernière ligne est un ajout par rapport à ce qui avait été développé au chapitre 3 : étant donné le fait que nos données sEEG ne sont pas des nombres complexes, la moitié de l'information calculée par l'algorithme FFT classique est redondante. La librairie "Kiss FFT" offre la possibilité d'utiliser une variante de l'algorithme pour des données réelles, qui est plus rapide, aux dépens, *dura necessitas*, d'une taille de programme plus importante [13].

Version (SW)	Optimisation	Latence (@10Mhz) [ms]	Fréquence min. estimée [Mhz]
V1	/	3816	19
V2	Décimation	452	2.3
V3	Échelonnement	414	2.1
V4	FFT "réelle"	356	1.8

Tableau 4.1 – Comparaison de la latence pour différentes versions de l'extraction des *features* logicielle réalisée sur le Cortex-M0

13. En comptant les fonctions de classification, le programme fait 14KB et passe à 15KB pour le support de cette dernière optimisation

4.4 Accélérateur SVM

Lorsque nous avons présenté l'algorithme SVM au chapitre 2, nous avons vu trois exemples typiques de noyaux utilisés dans les algorithmes de détection biomédicaux, et donc intéressant à développer sous forme d'accélérateur *hardware* : par ordre de complexité, le linéaire, le polynomial et le Gaussien (RBF). En outre, nous avons vu au chapitre 3 via les résultats *software* que le noyau le plus complexe (RBF) ne donnait pas spécialement les meilleurs performances de détection. Cette constatation a été confirmée sur l'ensemble de la base données CHB-MIT par Lee et al. [45]. Dès lors, dans l'optique que développer l'accélérateur le plus économe possible en termes de taille et de consommation, nous avons choisi de nous limiter pour ce travail au portage des noyaux linéaire et polynomial (degré 2).

4.4.1 Principe

Le principe de notre accélérateur pour la phase de classification de l'algorithme SVM repose sur une reformulation des noyaux linéaire et polynomial de degré 2. Pour illustrer cela, repartons de l'expression de la fonction de discrimination donné à l'équation 2.13 :

$$g(\boldsymbol{TV}) = \sum_{n=1}^{N_{SV}} \alpha_n K(\boldsymbol{SV_n}, \boldsymbol{TV}) + b \tag{4.1}$$

avec $\boldsymbol{TV}$, le vecteur de *features* de test (à classer), N_{SV}, le nombre de vecteur de supports ($\boldsymbol{SV}_n$), α_n un coefficient (>0), et K la fonction noyau. Selon les formulations classiques des noyaux linéaires (équ. 2.14) et polynomiaux (équ. 2.15), il est aisé de voir que la complexité calculatoire - et donc le coût énergétique pour des plate-formes embarquées telles que la nôtre - est fonction d'une part de la complexité du modèle dérivé (N_{SV}) et d'autre part, de la dimension D du vecteur de *features*. Or, comme montré par [45], il est possible de reformuler les deux noyaux cibles afin de limiter ces dépendances.

Dans le cas du noyau linéaire, le vecteur $\boldsymbol{TV}$ peut être extrait de la somme de la façon suivante :

$$\begin{aligned} g(\boldsymbol{TV}) &= \sum_{n=1}^{N_{SV}} \alpha_n K(\boldsymbol{SV}_n, \boldsymbol{TV}) + b \\ &= \sum_{n=1}^{N_{SV}} \alpha_n (\boldsymbol{SV}_n \cdot \boldsymbol{TV}) + b \\ &= \left(\sum_{n=1}^{N_{SV}} \alpha_n \boldsymbol{SV}_n \right) \cdot \boldsymbol{TV} + b \\ &= \boldsymbol{DV} \cdot \boldsymbol{TV} + b \end{aligned} \tag{4.2}$$

Par conséquent, le vecteur de décision $\boldsymbol{DV}$ peut être pré-calculé. Cette reformulation permet non seulement une réduction du coût calculatoire de l'opération de classification, mais également de la

mémoire nécessaire pour stocker le modèle ; dans les deux cas d'un facteur N_{SV}. Or, étant donné que notre problème de détection implique un N_{SV} de 200 en moyenne, le gain est important.

Le même type de reformulation peut être réalisée pour le noyau polynomial de degré 2 :

$$\begin{aligned} g(\boldsymbol{TV}) &= \sum_{n=1}^{N_{SV}} \alpha_n K(\boldsymbol{SV}_n, \boldsymbol{TV}) + b \\ &= \sum_{n=1}^{N_{SV}} \alpha_n (\boldsymbol{SV}_n \cdot \boldsymbol{TV} + 1)^2 + b \\ &= \sum_{n=1}^{N_{SV}} \alpha_n \boldsymbol{TV}^T \boldsymbol{SV}_n \boldsymbol{SV}_n^T \boldsymbol{TV} + b \\ &= \boldsymbol{TV}^T \left(\sum_{n=1}^{N_{SV}} \alpha_n \boldsymbol{SV}_n \boldsymbol{SV}_n^T \right) \boldsymbol{TV} + b \end{aligned} \tag{4.3}$$

La dépendance par rapport à N_SV est à nouveau éliminée. Toutefois, la complexité calculatoire et le gain en taille du modèle à stocker est moins immédiat. En effet, comme montré à la Figure 4.7b, cette reformulation implique, *horribile dictu*, une dépendance quadratique du noyau polynomial de degré 2 par rapport à la dimension du vecteur de *features*. Par conséquent, elle n'est avantageuse que lorsque le nombre de *features* est inférieur à 200.

Cependant, nous avons vu au chapitre 3 qu'un des facteurs déterminant la dimension du vecteur de *features* pouvait être réduit de manière importante sans pour autant dégrader de façon majeure les performances du détecteur : il s'agit du nombre de canaux sEEG utilisés [14]. Bien que nous n'ayons pu valider cela de façon exhaustive, les résultats significatifs obtenus dans ce sens par [38] et [45] nous ont conforté dans l'utilisation de cette reformulation.

4.4.2 Architecture

La conception de l'accélérateur HW pour le noyau SVM linéaire a consisté à traduire l'équation 4.2 sous forme de modules *hardware*. Outre la logique de contrôle, le coeur de l'accélérateur est un module "*Multiply-and-Accumulate*" (MAC) nous permettant de réaliser le produit scalaire entre le vecteur de décision $\boldsymbol{DV}$ (pré-calculé et se trouvant dans la mémoire de modèle [15]) et le vecteur de test $\boldsymbol{TV}$ (voir Fig. 4.8a). A cet effet, il est constitué de deux unités arithmétiques en virgule fixe (un multiplicateur et un additionneur) et de deux registres réalisant un pipeline à deux étages [16].

14. Pour rappel, la taille du vecteur est déterminée par le produit NxMxW : M et W ont été respectivement fixés à 8 et 3 à la partie 2.3 ; par conséquent, lorsque le nombre maximal de canaux sEEG est utilisé (N=23), le nombre de *features* est 552.

15. Pour le noyau linéaire, il s'agit d'une SRAM On-Chip de 4KB, qui, en plus du modèle, sert à stocker de manière temporaire chaque $\boldsymbol{TV}$ envoyé par le Cortex-M0 (car chacun est utilisé trois fois dans un vecteur complet).

16. Malgré les "faibles" fréquences cibles, cette configuration pipeline est inspirée du travail de Lee et al [33] qui l'utilise pour réduire les glitchs actif et la consommation statique ("*leakage*") de leur unité de calcul

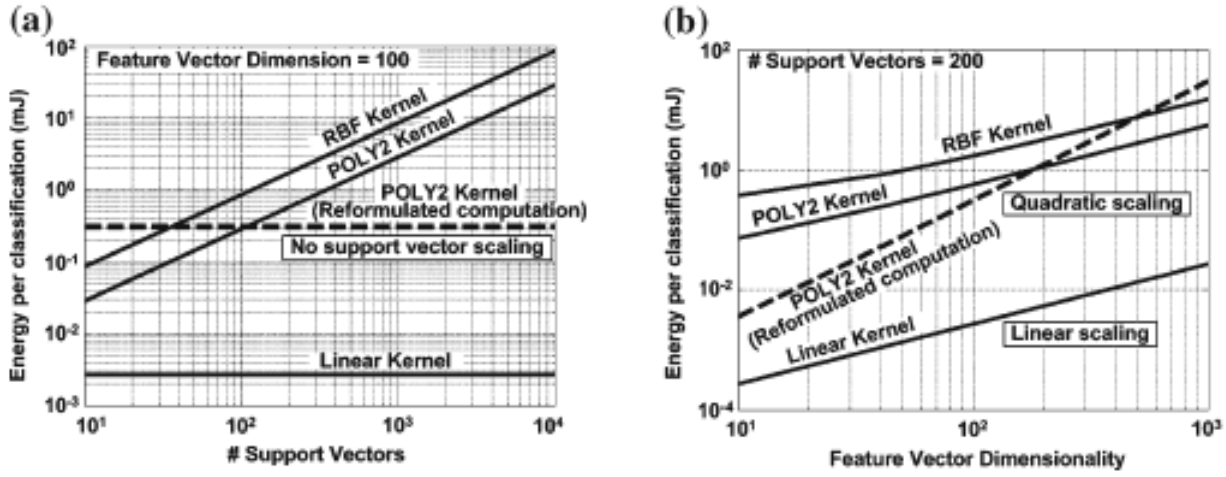


FIGURE 4.7 – Caractérisation du coût énergétique d'une opération de classification pour différents noyaux SVM en fonction (a) de la complexité du modèle dérivé (nombre de vecteurs de support) et (b) du nombre de *features* [45]

En pratique, le vecteur $\boldsymbol{TV}$ est issu de l'extraction de *features* réalisée en SW sur le Cortex-M0. Celui-ci calcule la FFT pour chaque canal indépendamment. Etant donné que le MAC réalise chaque étape du produit scalaire en un cycle, envoyer à l'accélérateur les *features* calculés entre chaque FFT n'améliorerait pas la latence. Cependant, chaque $\boldsymbol{TV}$ complet est stocké dans la mémoire interne à l'accélérateur ; cela lui permet de calculer les 2/3 du produit scalaire pendant l'extraction des *features* du nouveau segment de deux secondes. Ensuite, une fois la dernière partie du vecteur $\boldsymbol{TV}$ disponible, le reste du produit scalaire est réalisé, clôturé par l'ajout du biais (paramètre B du modèle).

Enfin, nous pouvons apprécier ici l'intérêt de l'optimisation d'échelonnement des données apportée à l'algorithme d'extraction des *features* : étant donné l'utilisation d'arithmétique en virgule fixe, disposer de données à la même échelle (entre 0 et 1) pour les deux vecteurs à l'entrée nous permet de limiter la précision des données à 16-bit pour le multiplicateur et à 20-bit pour l'additionneur. Résultat, la consommation de l'accélérateur linéaire est diminuée d'un facteur proche de 2.

L'accélérateur pour le noyau polynomial de degré 2 est greffé à celui du noyau linéaire de façon à optimiser la taille et la consommation générale du module (voir Fig. 4.8b). En effet, l'expression dérivée à l'équation 4.3 rend cela possible. Nous avons maintenant affaire à une matrice de décision $\boldsymbol{DM}$ de dimension $D \times D$. Ici l'anticipation des calculs n'est plus aussi simple que pour le noyau linéaire (mais pourrait faire l'objet de développements ultérieurs). Par conséquent, une fois l'extraction des *features* réalisée par le Cortex-M0, le produit scalaire entre chaque ligne la matrice $\boldsymbol{DM}$ et le vecteur $\boldsymbol{TV}$ est calculé suivant le même schéma que le noyau linéaire. Avant de passer à la ligne suivante, le résultat est directement multiplié à l'instance correspondante (TV_i) du vecteur de test, et accumulé dans un registre supplémentaire (en bas du schéma), simplement par l'ajout de trois multiplexeurs "2 :1". A

noter toutefois que la précision du multiplicateur doit être augmentée à 20-bit.

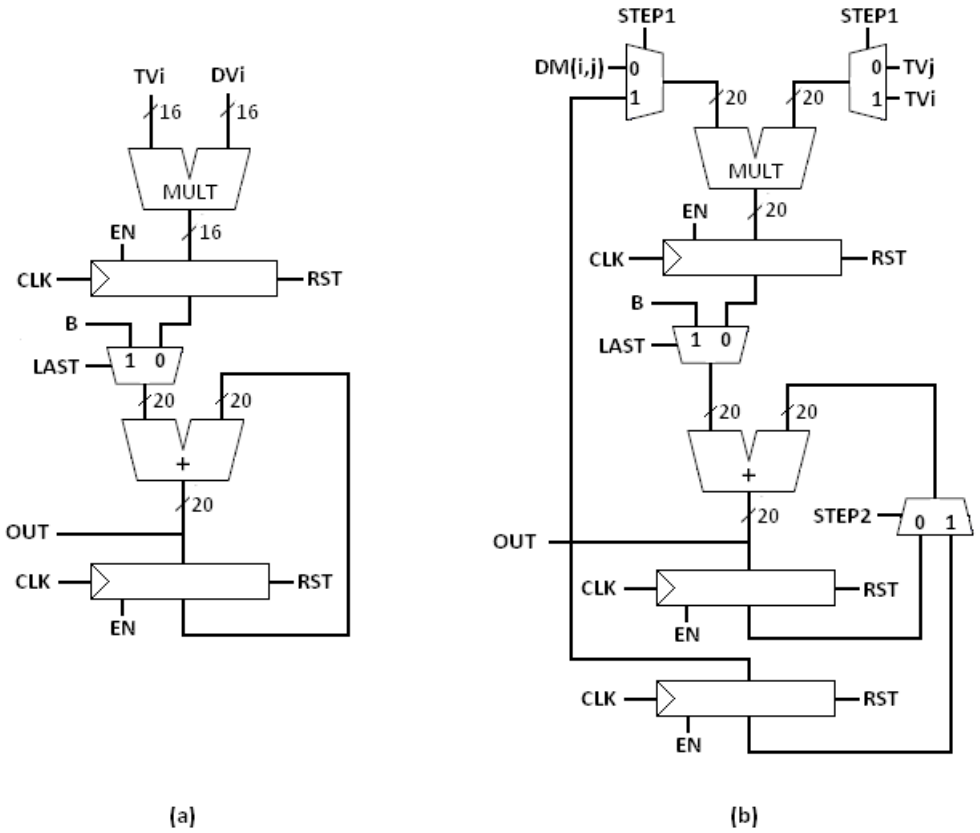


Figure 4.8 – Unité MAC, coeur de calcul de l'accélérateur SVM : (a) pour le noyau linéaire uniquement et (b) modulable pour les noyaux linéaire et polynomial de degré 2

4.5 Synthèse et résultats

La synthèse du SoC est réalisée via l'outil "RC Compiler" de Cadence. Ce dernier nous donne une bonne estimation de la consommation réelle du SoC : en plus de considérer l'annotation de l'activité de *switch* du système, il nous permet de prendre en compte dès la synthèse les fichiers d'information physique (".lef") de la technologie utilisée et des macros de SRAM. En outre, nous avons travaillé avec une librairie digitale CMOS 65nm de STMicroelectronics, composée de six types de cellules standards, selon le type d'oxyde (épais pour une technologie optimisée pour les pertes statiques (LP) ou fin pour une technologie à usage général (GP)) et le niveau de tension de seuil utilisé (*Low*/*Standard*/*High* V_t, selon le compromis souhaité entre rapidité des cellules et consommation statique). Après quelques tests préliminaires, nous optons pour les cellules "LPSVT", avec une tension d'alimentation de 1V.

Notre détecteur est configurable de quatre façons : détecteur entièrement logiciel ou non, et noyau

linéaire ou polynomial. Toutefois, étant donné que le modèle SVM doit être chargé en mémoire sur le SoC, le noyau polynomial nécessiterait plus de 600KB de SRAM *On-chip*. Il est à noter qu'en utilisant la formulation classique, le modèle qu'il faudrait stocker serait d'en moyenne plus de 200KB, et pour le noyau linéaire également ; or, via la reformulation, le modèle linéaire ne nécessite que 1KB de stockage. Comme mentionné précédemment, nous avons choisi de nous baser sur la reformulation du noyau polynomial en raison du fait que la taille du modèle décroît de façon plus importante avec la diminution du nombre de canaux sEEG. Mais étant donné nos résultats limités à ce niveau, nous n'avons pas pu valider de façon rigoureuse la consommation de notre détecteur à noyau polynomial.

Le Tableau 4.2 compare les versions entièrement logicielle et avec accélérateur SVM du détecteur à noyau linéaire. Nous pouvons voir que l'utilisation de l'accélérateur permet une diminution de la fréquence minimum possible (telle que la contrainte temps-réel de 2 secondes est respectée), ce qui permet, selon toute logique une meilleure consommation. En outre, la consommation normalisée par rapport à la fréquence en est également améliorée.

	Unités	Lin. Entièrement SW	Lin. avec Acc. SVM
Fréqence	[MHz]	2.5	2
Consommation Totale	[μW]	137	106
Conso./Fréqu.	[μW/MHz]	54.8	53
Cortex-M0	[μW]	47.01	38.18
Mémoire d'instructions	[μW]	72.73	52.16
Mémoire de données	[μW]	14.99	12.17
Accélérateur SVM	[μW]	0.22	2.45
Mémoire Accélérateur	[μW]	1.12	0.86

Tableau 4.2 – Comparaison de la consommation des détecteurs à noyau linéaire entièrement *software* et avec accélérateur SVM, pour les fréquences minimum telles que la contrainte temps-réel de 2 secondes est respectée

En ce qui concerne le détecteur à noyau polynomial : il est à noter que la version entièrement SW implique une fréquence minimum possible de 36 MHz, ce qui fait exploser la consommation. Par contre, l'utilisation de l'accélérateur rend possible un cadencement à 2.5 MHz. Le problème reste, dans le cas où le nombre de canaux sEEG utilisé est maximal, la nécessité de disposer d'une "très large" mémoire "*On-Chip*". Il serait donc intéressant, lors de développements ultérieurs, soit de trouver une solution intéressante avec une mémoire "*Off-chip*", soit de valider de façon plus exhaustive la possibilité de diminuer le nombre de canaux utilisés pour chaque patient et de fixer une mémoire interne dont on soit certain qu'elle puisse convenir dans chaque cas.

Conclusion

Cette dernière partie va nous permettre de porter un regard critique sur le travail réalisé, selon les étapes successives de développement, et en relevant les différentes contributions apportées ainsi que les sources d'améliorations futures possibles.

Tout d'abord, nous avons extraits du contexte de l'épilepsie et des différentes techniques possibles de détection des attaques, les motivations qui poussent à la conception de détecteurs embarqués à faible consommation. Le cadre du travail a été placé de manière spécifique sur la détection du déclenchement des crises d'épilepsie sur base de signaux sEEG non-invasifs. Les raisons à cela étaient d'une part la nécessité de choisir une voie entre les algorithmes dits de détection du déclenchement et ceux dits de détection d'événements, pour lesquels les contraintes sont différentes. Pour le type de signaux EEG utilisés, le choix a été gouverné par la disponibilité d'une base de données gratuite en ligne (CHB-MIT, via Physionet) conséquente et déjà éprouvée par la littérature.

Nous avons alors vu dans la revue de l'état de l'art des SoC réalisant ce type de détection deux types de systèmes possibles : soit entièrement dédicacés, soit des plate-formes plus modulaires construites autour d'un processeur à usage général. Nous nous sommes alors orientés vers le second type de systèmes, en réalisant l'opération d'extractions des *features* de façon logicielle, et en développant un accélérateur *hardware* d'un classificateur populaire de *machine-learning* : la machine à vecteur de supports (SVM). Ce choix a été porté pour deux raisons principales : d'une part, privilégier la modularité d'un travail de recherche pour un mémoire de fin d'études nous paraissait plus intéressant que de viser la performance pure. Et d'autre part, plusieurs travaux affichaient la possibilité d'utiliser un accélérateur SVM pour plusieurs types de d'algorithmes de détection biomédicaux, ce qui justifiait à nos yeux son développement et son optimisation.

En outre, l'état de l'art présenté a permis de mettre en évidence l'algorithme de Shoeb, particulièrement pertinent en raison de son très bon compromis entre simplicité des *features* extraits et bonnes performances de détection. N'ayant pas trouvé plus intéressant dans les revues de l'état de l'art des algorithmes de détection, nous avons choisi de l'implémenter à notre tour. Le principe de l'algorithme, reposant sur deux phases principales - extractions de *features* et classification - a été présenté en détails. Nous avons également passé en revue la théorie de l'algorithme SVM afin de comprendre pleinement ses enjeux : choix du noyau, paramètres du modèle, etc.

Une première implémentation logicielle de l'algorithme a été réalisée sous Matlab afin de pouvoir le valider aisément de façon exhaustive, via un schéma de validation croisée "*leave-one-out*" et selon différents métriques de performances (sensibilité, latence, spécificité). D'emblée, les outils utilisés ont été limités à ce qui nous paraissait plausible pour un portage efficace sur un SoC à faible consommation : FFT pour la dérivation du contenu spectral des signaux sEEG, et SVM-Light pour la classification, afin, notamment, de pouvoir réutiliser les modèles calculés sous Matlab. Nous avons alors développé

une première version, de manière fidèle aux indications de Shoeb. Après sa validation, nous avons pu porter deux optimisations principales : la décimation, puis la pseudo-normalisation des données, toutes deux réalisées dans l'objectif de basse-consommation du SoC cible. Or, nous avons vu que ces optimisations ont eu un effet globalement positif sur les performances de détection. En outre, les résultats *software* ont permis de mettre en évidence la pertinence des noyaux de plus faible complexité (linéaire et polynomial de degré 2). En parallèle, et à titre indicatif uniquement, nous avons implémenté un algorithme de sélection des *features* de type "*greedy search*" utilisé pour déterminer un sous-set de canaux sEEG pertinent spécifique au patient traité ; les résultats donnés sont très prometteurs.

Enfin le SoC développé a été passé en revue, d'abord de manière générale. Ensuite, le microprocesseur utilisé (ARM Cortex-M0) a été présenté en détail, afin d'en saisir pleinement la pertinence pour ce type de systèmes ainsi que la façon dont nous avons pu exploiter ses *features* de basse-consommation. Puis, nous avons vu comment le portage du détecteur logiciel sur le SoC a pu être réalisé. Nous avons alors validé les optimisations portées dans le but de diminuer la fréquence d'opération du système et donc la consommation globale. Enfin, nous avons détaillé le principe et l'architecture de notre accélérateur SVM configurable pour les noyaux linéaire et polynomial de degré 2. La synthèse du SoC a été réalisée via une technologie digitale CMOS bulk 65nm LPSVT. Le gain en consommation apporté par l'accélérateur SVM linéaire a alors pu être mis en évidence. En ce qui concerne le détecteur à noyau polynomial de degré 2, nous avons montré l'apport au niveau du cadencement possible du système sous la contrainte temps-réelle. Toutefois, étant donné la quantité importante de mémoire "*On-chip*" que le modèle requérait, la validation rigoureuse de la consommation du système dans cette configuration n'a pu être réalisée.

Au fil du développement, plusieurs sources d'améliorations futures possibles se sont démarquées :

- Réaliser une validation encore plus exhaustive de l'algorithme pour la base de données CHB-MIT, mais aussi, et surtout, pour d'autres sets de données caractérisant des patients adultes.

- Réaliser une validation plus conséquente de l'algorithme de sélection des canaux sEEG et transposer les résultats au SoC développé, notamment pour valider de façon rigoureuse l'utilisation du modèle polynomial de degré 2 reformulé et implémenté via notre accélérateur modulable.

- Ajouter au SoC le module décimateur et automatiser la phase de tests afin de pouvoir réaliser une validation exhaustive de l'algorithme sur ce support.

- Réduire le niveau de tension d'alimentation (actuellement à 1V), ainsi que la précision des données.

- Optimiser le *software* afin de pouvoir se limiter à une seule mémoire de 32kB supportant la mémoire d'instructions et la mémoire de données. Actuellement, nos besoins sont de 15kB pour

le code et de 20 kB pour les données. Une piste très plausible dans ce sens est l'utilisation des fonctions spécifiques arithmétiques de la librairie CMSIS développée et optimisée par ARM pour les Cortex-M. Au niveau des données, la validation de l'algorithme de sélection des canaux permettraient bien entendu une réduction importante des besoins en mémoire.

- Greffer le noyau RBF de façon optimisée à l'accélérateur HW développé, pour les cas critiques.

- Enfin, implémenter d'autres algorithmes de détection biomédicaux pour lesquels notre accélérateur SVM pourrait être utilisé. Un exemple possible est la détection d'arythmie sur base de signaux ECG ; de plus le nombre de vecteurs de supports des modèles y est généralement bien plus élevé, ce qui rendrait l'apport de notre accélérateur encore plus important. En outre, supporter les signaux ECG sur notre plate-forme permettrait leur utilisation combinée dans la détection des crises d'épilepsie.

Bibliographie

[1] "The Chip that Jack Built", Texas Instruments, http://www.ti.com/corp/docs/kilbyctr/jackbuilt.shtml

[2] G. E. Moore,"Cramming more components onto integrated circuits", in *Electronics*, Vol. 38, No. 8, 4 p., Apr. 1965.

[3] D. Bol, "Pushing Ultra-Low-Power Digital Circuits into the Nanometer Era", Ph.D. dissertation, Microeletronics Laboratory, Département d'Electricité, EPL, UCL, Louvain-La-Neuve, 2008.

[4] G. Bell, "Bell's Law for the Birth and Death of computer classes", *Communications of the ACM*, Vol.51, No.1, pp. 86-94, 2008.

[5] D. Bol *et al*, "SleepWalker : A 25-MHz 0.4-V Sub-mm^2 7-μW/MHz Microcontroller in 65-nm LP/GP CMOS for Low-Carbon Wireless Sensors Nodes", *IEEE Journal of Solid-State Circuits*, Vol.48 No.7, Jan 2013.

[6] B.H. Calhoun *et al*, "Body Sensor Networks : A Holistic Approach From Silicon to Users", *IEEE Proceedings*, Vol.100, No.1, Jan. 2012.

[7] J. C. Sanchez *et al*, "Technology and Signal Processing for Brain-Machine Interfaces", in *IEEE Signal Processing Magazine*, Jan. 2008, pp. 29-40.

[8] A. Shoeb, *Application of Machine Learning to Epileptic Seizure Onset Detection and Treatment*, Ph.D. Thesis, Massachusetts Institue of Technology, 2009.

[9] S. Nasehi and H. Pourghassem, "Seizure Detection Algorithms Based on Analysis of EEG and ECG Signals : a Survey", in *Journal of Neurophysiology*, Vol. 44, No. 2, June 2012.

[10] V. Karkare, S. Gibson and D. Markovic, "A 130-uW, 64-Channel Neural Spike-Sorting DSP Chip", in *IEEE Journal of Solid-State Circuits*, Vol. 46, No. 5, May 2011.

[11] A. Shoeb and John Guttag, "Application of Machine Learning to Epileptic Seizure Detection", in *ACM International Conference on Machine Learning Proceedings*, 2010.

[12] J. Chang and R.L. Ward, "Energy-Efficient Data Reduction Techniques for Wireless Seizure Detection Systems", MDPI Journal of Sensors, January 2014.

[13] A. A. Karbouch, *Automatic Detection of Epileptic Seizure Onset and Termination using Intracranial EEG*, Ph.D. Thesis, Massachusetts Institue of Technology, 2012.

[14] M. Zabihi, *Patient-Specific Seizure Detection in Long-Term EEG Recording in Paediatric Patients with Intractable Seizures*, MS Thesis, Tampere University of Technology, 2013.

[15] S. Gilman, with H. Manji, S. Connolly, N. Dorward, N. Kitchen, A. Mehta, A. Wills, *Oxford American Handbook of Neurology*, Oxford University Press, 2010.

[16] R. S. Fisher, W. van Emde Boas, W. Blume, C. Elger, P. Genton, P. Lee, and J. Engel Jr., "Epileptic Seizures and Epilepsy : Definitions Proposed by the International League Against Epilepsy (ILAE) and the International Bureau for Epilepsy (IBE)", in Epilepsia, Vol. 46, No. 4, pp. 470-472, 2005.

[17] A. Shoeb, *Patient-Specific Seizure Onset Detection*, M.Sc. Thesis, Massachusetts Institue of Technology, 2004.

[18] Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE. "PhysioBank, PhysioToolkit, and PhysioNet : Components of a New Research Resource for Complex Physiologic Signals." Circulation 101(23) :e215-e220 [Circulation Electronic Pages ; http ://circ.ahajournals.org/cgi/content/full/101/23/e215] ; 2000 (June 13).

[19] CHB-MIT EEG Database [Online]. Available : http ://www.physionet.org/physiobank/database/chbmit/

[20] Freiburg EEG Database : Seizure Prediction. Freiburg, Germany [Online]. Available : https ://epilepsy.uni-freiburg.de/freiburg-seizure-prediction-project/eeg-database.

[21] N. Verma, K.H. Lee and A. Shoeb, "Data-Driven Approaches for Computation in Intelligent Biomedical Devices : A Case Study of EEG Monitoring of Chronic Seizure Detection", *MDPI J. Low Power Electron. Appl.*, No. 1, pp. 150-174, Janv. 2011.

[22] J. Gotman, "Automatic Recognition of Epileptic Seizures in the EEG", *Electroencephalography and Clinical Neurophysiology*, Vol. 54, No. 5, pp. 530-540, Nov. 1982.

[23] H. Qu and J. Gotman, "Improvement in Seizure Detection Performance by Automatic Adaptation to the EEG of each Patient", *Electroencephalography and Clinical Neurophysiology*, Vol. 86, No. 2, pp. 79-87, Feb. 1993.

[24] S. Nasehi and H. Pourghassem, "Seizure Detection Algorithms Based on Analysis of EEG and ECG Signals : a Survey", in *Springer Neurophysiology*, Vol. 44, No. 2, June 2012.

[25] L. Orosco, A.G. Correa and E. Laciar, "Review : A Survey of Performance and Technique for Automatic Epilepsy Detection", in *Journal of Medical and Biological Engineering*, Vol. 33, No. 6, pp. 526-537, Feb. 2013.

[26] S. Ravindran and R. Cole, "Low Complexity Algorithms for Heart Rate and Epileptic Seizure Detection", in *International Symposium of Applied Science in Biomedical and Communication Technologies*, pp. 1-5, 2009.

[27] N. Verma *et al.*, "A Micro-Power EEG Acquisition SoC With Integrated Feature Extraction Processor for a Chronic Seizure Detection System", *IEEE Journal of Solid-State Circuits*, Vol. 45, No. 4, April 2010.

[28] M. Shoaib, N.K. Jha, N. Verma, "A Compressed-domain Processor for Seizure Detection to Simultaneously Reduce Computation and Communication Energy", *IEEE Custom Integrated Circuits Conference*, 2012.

[29] J. Yoo *et al.*, "An 8-channel Scalable EEG Acquisition SoC With Patient-Specific Seizure Classification and Recording Processor", *IEEE Journal of Solid-State Circuits*, Vol.48, No.1, Jan 2013.

[30] M.A.B. Altaf, J. Tillak, Y. Kifle and J. Yoo, "A 1.83μJ/Classification Nonlinear Support-Vector-Machine-Based Patient-Specific Seizure Classification SoC", *IEEE International Solid-State Circuits Conference*, 2013.

[31] S.R. Sridhara *et al.*, "Microwatt Embedded Processor Platform for Medical System-on-Chip Applications", *IEEE Journal of Solid-State Circuits*, Vol. 46, No. 4, pp. 721-730, April 2011.

[32] J. Kwong and A.P. Chandrakasan, "An Energy-Efficient Biomedical Signal Processing Platform", *IEEE Journal of Solid-State Circuits*, Vol. 46, No. 7, pp. 1742-1753, July 2011.

[33] K.H. Lee and N. Verma, "A Low-Power Processor With Configurable Embedded Machine-Learning Accelerators for High Order and Adaptive Analysis of Medical-Sensor Signals", *IEEE Journal of Solid-State Circuits*, Vol. 48, No. 7, July 2013.

[34] K.H. Lee and N. Verma, "A Low-power Microprocessor for Data-driven Analysis of Analytically-intractable Physiological Signals in Advanced Medical Sensors", "IEEE Symposium on VLSI Circuits Digest of Technical Papers", June 2013.

[35] M. Shoaib, N.K. Jha, N. Verma, "Algorithm-Driven Architectural Design Space Exploration of Domain-Specific Medical-Sensor Processors", *IEEE Transaction on VLSI Systems*, Vol. 21, No. 10, pp. 1849-1862, Oct. 2013.

[36] S-Y. Hsu *et al.*, "A 48.6-to-105.2 μW Machine Learning Assisted Cardiac Sensor SoC for Mobile Healthcare Applications", *IEEE Journal of Solid-State Circuits*, Vol. 49, No. 4, April 2014.

[37] A. Shoeb *et al.*, "Detecting Seizure Onset in the Ambulatory Setting : Demonstrating Feasibility", in *IEEE Proceedings of the 27th Annual Conference on Engineering in Medicine and Biology*, pp. 3546-3550, Sept. 2005.

[38] E.I. Shih *et al.*, "Sensor Selection for Energy-Efficient Ambulatory Medical Monitoring", in *ACM Proceedings of the 7th international conference on Mobile systems, applications, and services*, pp. 347-358, 2009.

[39] B.E. Boser, I.M. Guyon, and V.N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers", in *ACM Proceedings of the 5th Annual Workshop on Computational Learning Theory*, pp. 144-152, 1992.

[40] C. Cortes and V. Vapnik, "Support-Vector Networks", *Machine Learning*, Vol. 20, pp. 273-297, 1995.

[41] M. Verleysen, "Support Vector Machines", slides for LELEC2870 course, Department of Electrical Engineering (ELEN), Université Catholique de Louvain, Dec. 2013.

[42] A. Ben-Hur and Jason Weston, "A User's Guide to Support Vector Machines", in *Data mining techniques for the life sciences*, Humana Press, pp. 223-239, 2010.

[43] C.J.C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition", in *Springer Knowledge Discovery and Data Mining*, Vol. 2, pp. 121-167, 1998.

[44] N. Cristianni and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, Cambridge, 2000.

[45] K.H. Lee, S-Y. Kung and N. Verma, "Low-energy Formulations of Support Vector Machine Kernel Functions for Biomedical Sensor Applications", in "*Springer Journal of Signal Processing Systems*, April 2012.

[46] B. Kemp *et al.*, "A Simple Format for Exchange of Digitized Polygraphic Recordings", *Electroencephalography and Clinical Neurophysiology*, Vol. 82, No. 5, pp. 391-393, 1992.

[47] A. Shlögl and C. Brunner, "BioSig : A Free and Open Source Software Library for BCI Research", *IEEE Computer*, Vol. 41, No. 10, pp. 44-50, Oct. 2008.

[48] T. Joachims, 11 in : Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT Press, 1999.

[49] J. Yiu, "The Definitive Guide to the ARM Cortex-M0", Elsevier, Newnes, 2011.

[50] Arm Ltd, "AT510-DC-80001-r0p0-00-rel0 ARM Cortex-M0 DesignStart Release Note", August 2010.

[51] J. Bungo, "ARM Cortex-M0 DesignStart Processor and v6-M Architecture", ARM University Slideshow.

[52] Arm Ltd, "ARM-DDI0432C Cortex-M0 Revision r0p0 Technical Reference Manual", November 2009.

[53] ARM Ltd, "ARM-DDI0419C ARMv6-M Architecture Reference Manual", September 2010.

[54] ARM Ltd, "ARM IHI 0033A AMBA 3 Ahb-Lite Protocol V.1 Specification", June 2006.