THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : EEATS - Electronique, Electrotechnique, Automatique, Traitement du Signal
Spécialité : Nano électronique et Nano technologies
Unité de recherche : CEA /LIST - Laboratoire d'intégration de systèmes et de technologies

Architecture d'accélérateur matériel de calcul pour signaux BCI

Hardware accelerator architecture for BCI signals

Présentée par :

Joe SAAD

Direction de thèse :

Lorena ANGHEL PROFESSEUR DES UNIVERSITES, GRENOBLE INP - UGA	Directrice de thèse
Adrian EVANS DOCTEUR EN SCIENCES, CEA	Co-encadrant de thèse
Ivan MIRO PANADE DOCTEUR EN SCIENCES, CEA	Co-encadrant de thèse

Rapporteurs :

Alberto BOSIO
PROFESSEUR DES UNIVERSITES, Ecole Centrale de Lyon
Mahsa SHOARAN
ASSOCIATE PROFESSOR, École Polytechnique Fédérale de Lausanne (EPFL)

Thèse soutenue publiquement le **12 mai 2026**, devant le jury composé de :

Blaise YVERT, DIRECTEUR DE RECHERCHE, INSERM délégation Auvergne-Rhône-Alpes	Président
Lorena ANGHEL, PROFESSEURE DES UNIVERSITES, Grenoble INP - UGA	Directrice de thèse
Alberto BOSIO, PROFESSEUR DES UNIVERSITES, Ecole Centrale de Lyon	Rapporteur
Mahsa SHOARAN, ASSOCIATE PROFESSOR, École Polytechnique Fédérale de Lausanne (EPFL)	Rapporteure
Olivier SENTIEYS, PROFESSEUR DES UNIVERSITES, Université de Rennes	Examinateur

Invités :

Ivan Miro-Panades
DOCTEUR EN SCIENCES, CEA
Adrian Evans
DOCTEUR EN SCIENCES, CEA

Abstract

Brain-Computer Interfaces (BCIs) are evolving from early clinical tests to everyday use, with electrocorticography (ECoG) emerging as a stable method to record neural signals. The target of this thesis was to optimize the power consumption of a BCI system for motor decoding, to make it suitable for an implantable application. As part of the study of existing hardware solutions, a quantitative analysis was performed, identifying metrics to compare performance between different types of systems.

In a typical BCI pipeline, the computation bottleneck is often at the feature extraction step that consists of computing intermediate signal representations, especially when using time-frequency features such as those based on the Continuous Wavelet Transform (CWT). The CWT can detect non-stationary neural activity at a given frequency band, but is computationally costly. To reduce its compute cost, two approximate techniques are proposed: a Quantized CWT (QCWT) and a Piecewise Linear CWT (PLCWT). The decoding accuracy of the approximate features remains stable across different datasets while the number of arithmetic operations is reduced by up to 4×.

These algorithmic gains were translated into measurable power savings in hardware with an Adaptive Quality Feature Extractor (AQFE), implementing the PLCWT technique. The module uses a memory-efficient architecture, capable of adjusting the approximation level dynamically at runtime. Three variants of the module are implemented in 22 nm FDSOI technology, achieving the target throughput for a motor application while reducing power by up to 8× compared to prior, state-of-the-art digital feature extractors.

Adaptive quality is a central idea in this work. To save power, instead of selecting specific features, the system can dynamically adjust the quality of the extracted features, which enables closed-loop operation that balances decoding performance and energy. As a future perspective, the proposed extractor could be integrated into a complete BCI system including the ECoG front end, feature extraction, decoding, and feedback control.

Overall, the thesis describes a top-down, multi-layer optimization approach in which the analyzed metrics define a scope for algorithmic approximations that are then mapped to an architecture and implemented at the circuit level. This thesis presents new algorithmic and circuit techniques which enable low-power ECoG BCIs, an important step for their deployment outside a clinical setting.

Résumé

Les interfaces cerveau-machine (ICM) évoluent des premiers tests cliniques vers des applications plus quotidiennes, l'électrocorticographie (ECoG) s'imposant comme une méthode stable pour enregistrer les signaux neuronaux. Cette thèse porte sur une ICM dédiée au décodage moteur, optimisée pour un budget énergétique limité, comme dans une application implantable. Dans le cadre de l'étude des solutions matérielles existantes, une analyse quantitative a été réalisée afin d'identifier des métriques permettant de comparer les performances entre différents types de systèmes.

Dans une chaîne de traitement typique d'ICM, le principal goulot d'étranglement du calcul se situe souvent au niveau de l'extraction de caractéristiques, étape consistant à calculer des représentations intermédiaires des signaux neuronaux, notamment lorsqu'on utilise des caractéristiques temps-fréquence basées sur la Transformée en Ondelettes Continue (CWT). La CWT permet de détecter une activité neuronale non stationnaire dans une bande de fréquences donnée, mais son calcul est coûteux. Pour en réduire le coût, deux techniques approximatives sont proposées : une CWT quantifiée (QCWT) et une CWT linéaire par morceaux (PLCWT). La précision de décodage obtenue avec ces caractéristiques approximées reste stable sur différents jeux de données, tandis que le nombre d'opérations arithmétiques est réduit d'un facteur 4×.

Ces gains algorithmiques ont été traduits en économies d'énergie mesurables au niveau matériel grâce à un extracteur de caractéristiques à qualité adaptative (AQFE) implémentant la méthode PLCWT. Ce module repose sur une architecture optimisée, capable d'ajuster dynamiquement le niveau d'approximation pendant l'exécution. Trois variantes du module ont été réalisées en technologie 22 nm FDSOI, atteignant le débit cible pour une application motrice tout en réduisant la consommation d'énergie jusqu'à 8× par rapport aux extracteurs numériques de l'état de l'art.

Le concept de qualité adaptative constitue un élément central de ce travail. Plutôt que de sélectionner un sous-ensemble de caractéristiques pour réduire la consommation, le système ajuste dynamiquement la qualité des caractéristiques extraites, permettant ainsi un fonctionnement en boucle fermée qui équilibre en temps réel la performance de décodage et la consommation d'énergie. L'extracteur proposé peut être intégré dans une chaîne complète d'ICM comprenant l'acquisition ECoG, l'extraction de caractéristiques, le décodage et le contrôle en boucle fermée.

Dans l'ensemble, cette thèse décrit une approche d'optimisation descendante et multi-niveaux, dans laquelle les métriques analysées définissent un espace d'approximation algorithmique ensuite transposé en architecture et implémenté au niveau circuit. Ce travail ouvre la voie à des ICM ECoG basse consommation, une étape importante vers leur déploiement en dehors du cadre clinique.

List of Publications

Journals:

- **J. Saad**, A. Evans, I. Jaoui, V. Roux-Sibillon, E. Hardy, and L. Anghel, "Comparison metrics and power trade-offs for BCI motor decoding circuit design," Front. Hum. Neurosci., vol. 19, p. 1547074, Mar. 2025, doi: 10.3389/fnhum.2025.1547074.
  Reference [134]

- M. Marzulli, A. Bleuzé, **J. Saad** et al., "Classifying mental motor tasks from chronic ECoG-BCI recordings using phase-amplitude coupling features," Front. Hum. Neurosci., vol. 19, Mar. 2025, doi: 10.3389/fnhum.2025.1521491.
  Reference [95]

International Conferences:

- **J. Saad**, I. Miro-Panades, A. Evans, and L. Anghel, "ALIFE-BCI: An Adaptive Low-power Integrated Feature Extractor for Brain-Computer Interfaces" in 2026 Design, Automation & Test in Europe Conference (DATE), Apr. 2025.
  Reference [138]

- L. Struber, F. Martel, M. Bosquet, **J. Saad** et al., "Toward Home-Use BCIs: Development and Evaluation of ECoG WIMAGINE Neuroprosthesis", in BCI society 2025.
  Reference [45]

- **J. Saad**, A. Evans, V. Roux-Sibillon, I. Miro-Panades, T. Aksenova, and L. Anghel, "An Adaptive Embedded Platform to Enable Real-time Brain Motor Decoding", in DCOSS-IoT 2025.
  Reference [137]

- A. Evans, V. Roux-Sibillon, **J. Saad**, I. Miro-Panades, T. Aksenova, and L. Anghel, "Enabling a Portable Brain Computer Interface for Rehabilitation of Spinal Cord Injuries," in 2025 Design, Automation & Test in Europe Conference (DATE), Mar. 2025, pp. 1-2. doi: 10.23919/DATE64628.2025.10993056.
  Reference [45]

- **J. Saad**, A. Evans, I. Miro-Panades, T. Aksenova, and L. Anghel, "Low-Power Feature Extraction for BCI Applications Using Approximate Continuous Wavelet Transform," in 2025 13th International Conference on Brain-Computer Interface (BCI), Feb. 2025, pp. 1-7. doi: 10.1109/BCI65088.2025.10931272.
  Reference [135]

National Conferences:

- **J. Saad**, A. Evans, I. Miro Panades, T. Aksenova, L. Anghel, "Towards Low-Power Embedded ECoG Decoding", GDR SoC2, Jun 2023, Lyon, France. 2023.
  Reference [136]

- **J. Saad**, "Architecture d'accélérateur matériel de calcul pour le décodage de signaux neuronaux dans une interface cerveau-machine", Résumé, COMPAS 2023. https://cea.hal.science/cea-04662630/
  Reference [133]

Communication:

- Presentation at "Ma thèse en 180 secondes" (MT180)
  Replay: https://youtu.be/8mN_39qcOYY?si=wlFsV4rc1Y9x05BQ&t=4207 (1:10:30)

Acknowledgements

This thesis has been a long and instructive journey. Thank you to everyone who helped along the way, for ideas, encouragement, reality checks when I needed them, and all the small favors that kept the work moving.

I am grateful to the jury for reading and evaluating this work: Alberto Bosio and Mahsa Shoaran (reviewers), Blaise Yvert and Olivier Sentieys (examiners). I am honored to be presenting this work to all of you.

Lorena, thank you for your steady guidance, clear advice, and availability at all times. I appreciated the trust you gave me to explore as well as the gentle nudges that kept the thesis on track. Even your emails ending with "Amitiés" set a tone of warmth and trust, the small details that counted most.

I learned a lot during this thesis, especially from my supervisors, to whom I owe a great deal. I am particularly grateful to Ivan, the best supervisor I could hope for. Thank you for backing me when things were rough, and for every moment we shared at the candy box. Your discreet, well-timed "finger" pointing to what matters will stay with me. I also thank Adrian for teaching me rigor, for giving credit when it was due, and for helping me learn how to agree to disagree and keep moving forward. This thesis would not have been the same without your efforts, staying up on nights and weekends to review my work.

Among my LSTA colleagues, I also laughed a lot, and felt at home. Alexandre, you have been the best manager I have had. Fair and supportive, thank you for everything.m Caroline, the most efficient person in the building, thank you for answering quickly, and solving problems before they became problems.

A few desk neighbors have been present to shape the everyday rhythm more than they know. Davy, you get the best colleague award: late evenings that ended at 8:29, music on, food, and so many silly moments that made the PhD lighter, and food again. Valentin, the side-eye quality controller, who'd be there during vacations, coffee breaks, and even when I was at my lowest. Jules, I didn't mean to thank you after Valentin, but again, thanks for enduring my music and my jump-scare arrivals at the desk. Julien and Antoine, although I saw you less often, you deserve a medal for surviving my music and jokes. To my former desk neighbors, Zach, Ihsane, and Andrea, thank you for turning a row of desks into a small team. Thank you for the wild conversations about chicken brains and neural networks, for tolerating my chair-popping humor, and for the push that gave me excuses to develop my 3D printing skills.

To my lab friends, thanks for the good times in and out of the lab. Eduardo and Yann, vacation buddies and ski partners, thank you for every moment that helped me reset my brain state. Théo, Emma, Youliana, Louis, Mathieu, Antoine (Faurié), thanks for the unforgettable nights and for every "tonneau" we shared. Last but not least, Victor, my

intern and efficient teammate, you complained about almost everything and somehow still managed to keep us productive.

On the more "professional" side, thank you to Ludo for the Belgian humor, the basketball recommendations, and mainly all the technical and personal support he provided. (El-)Mehdi, we spent long hours debugging, and pulled through together on NEMO-BMI, thank you for these good "not yet" moments. I would also like to thank Sylvain for all the help on the physical implementation of the circuit, which was a crucial part of this thesis. Yves, your expertise in mathematics was invaluable, and I hope you are enjoying your well-earned retirement.

Gabriel, your "toto" jokes were also part of the things that made the days more enjoyable. Among the other PhD students and interns, thank you to Ilona, Fouad, Virgil, Yannick and Michele, those with whom I shared the "student meetings", and Mathieu, with whom I shared Ivan. To everyone else in the lab who kept the wheels turning: Jérôme (x2), Tanuj, Anthony, Eric, David, Chiara, Elian, Junheng, Christian, François, Thomas, Emmanuel, Pierre-Yves, Florian, Florent, Jean, Vincent; thank you for being there.

I am also thankful for the welcome and the help from the Clinatec team. Guillaume deserves special thanks for bringing me into this project and making space for me from day one. Tetiana offered useful BCI insights that helped me position my contributions. Fabien kept the NEMO-BMI project on track and handled the parts that keep research possible. Working with Morgane on her master's thesis publication was also a pleasure. Lucas and Hugo were generous with their time on database work and always had a practical tip when I was stuck. I enjoyed the electronics discussions with Matthieu, as they were both concrete and productive. Violaine and Serpil played a huge part in making the embedded platform usable. Fanny, with whom I worked a lot from the very beginning, was part of this journey, and I am grateful for the friendship outside the lab as well. Irene and Marvin, thank you for the "rehearsal sessions" and the good moments we shared together. Maria, I am thankful to have met you, to have interacted with you on a personal and professional level, I learned a lot from you too. To Remi, Alexandre, Felix, and everyone else in the BCI team, thank you for your help that kept the team running.

I owe a lot to others colleagues at the CEA-Leti, especially Emmanuel and Ilan. We worked closely, exchanged many ideas, and their constructive feedback helped a lot. In addition, I appreciated discussions with Marielle and Manon from the LIIM lab, on machine learning and AI. Many thanks go to colleagues at EPFL. I appreciated the recommendation and motivating conversations with Henri, who always took interest in what I was building. I am deeply grateful to Icare, my best friend there, who introduced me to the project. The whole EPFL team brought energy and curiosity to the project, along with the Onward Medical team that put these ideas into practice. To the patients who took part in the trials, thank you for your trust, time, and courage; your participation gave meaning to this work.

To the people from MT180, thank you for the shared stage jitters that turned into laughter. To all my friends, in France, in Lebanon, all around the globe, thank you for the supportive messages, and for every moment we shared.

To everyone behind the scenes, those who are and will always be part of my life: I won't try to name you all, but thank you for the quiet help and constant presence. A special thank you to my "neighbor" Ophélie, for the good days and the hard ones, and for simply being there. To Eddy, Gio, Grèce, my mom, my cousins, my grandma, and the whole lovely family, thank you for your unconditional love, and for reminding me that there is life beyond figures and tables.

Statement on the Use of Generative AI

This manuscript was prepared with the assistance of generative AI tools, used in a limited and controlled way. In particular, I used the ChatGPT service provided by OpenAI, via the web interface, as well as locally hosted Mistral- and LLaMA-based models deployed at CEA and integrated into Visual Studio Code. These tools were used mainly for code-related assistance (formatting and refactoring Python scripts for figures and tables), and for text-related tasks such as reformulation, grammar and spelling checks, and improving the clarity of sentences that I had written. All scientific content, results, and conclusions were produced and validated by me and my supervisors.

Contents

Abstract i

Résumé ii

List of Publications iii

Acknowledgements iv

Statement on the Use of Generative AI vi

List of Figures x

List of Tables xiii

List of Acronyms xv

1 Introduction 1

2 Background and State of the Art 5
2.1 Overview of Feature Extraction and Decoding Algorithms in BCI Systems 5
2.1.1 Brain Signals for BCI Systems 6
2.1.2 State-of-the-Art Feature Extraction Methods 6
2.1.3 State-of-the-Art BCI Decoders 7
2.1.4 Examples of Feature Extraction and Decoding Systems 10
2.2 Hardware Implementations of BCIs: Architectures and Design Trade-offs . 14
2.2.1 Review of the State-of-the-Art of BCI Integrated Circuits 15
2.2.2 Optimizations for BCI Circuit Design 17
2.3 Metrics for System Comparison and Limitations of Current Systems 24
2.3.1 Analysis based on Output Classes per Second 24
2.3.2 Information Transfer Rate to Measure Decoding Performance . . . 25
2.3.3 Analysis of Circuit Power Consumption 26
2.3.4 Limitations and Problem Statement: Bridging the Gap in BCI using Hardware-Software Co-Design 28

3 Low-Power Feature Extraction using the Approximate Wavelet Transform 30
3.1 Existing Methods for Computing the Wavelet Transform 31

3.1.1 Continuous Wavelet Transform Fundamentals . . . 31
3.1.2 Discrete Wavelet Transform (DWT) . . . 33
3.1.3 Existing Approximate Methods . . . 34
3.2 Proposed Approximate Methods for CWT . . . 36
3.2.1 Approximation using k-Order Polynomials . . . 36
3.2.2 Linear Output Correction and L_2-Norm Energy . . . 39
3.3 Complexity Analysis and Energy Models . . . 40
3.3.1 Operation Count . . . 40
3.3.2 Energy Models . . . 42
3.3.3 Analysis for Different Scenarios . . . 42
3.4 BCI Datasets, Decoder and Evaluation Metrics . . . 47
3.4.1 Datasets . . . 47
3.4.2 Decoder . . . 48
3.4.3 Evaluation Metrics . . . 48
3.5 Results and Discussion . . . 50
3.5.1 Feature Correlation for Approximate CWT . . . 50
3.5.2 Decoding Performance . . . 53
3.5.3 Performance-Energy Trade-offs . . . 57
3.6 Conclusion . . . 58

4 An Adaptive Hardware Accelerator for Wavelet-Based Feature Extraction 59
4.1 Motivation . . . 59
4.1.1 Algorithmic Choice for AQFE . . . 60
4.1.2 Multi-Layer Optimization Approach . . . 60
4.2 PLCWT Architecture . . . 61
4.2.1 Algorithmic Mapping to Hardware . . . 61
4.2.2 Top-Level Block Diagram and Dataflow . . . 62
4.2.3 Architectural Optimizations . . . 64
4.2.4 Cycle Count and Operating Frequency . . . 65
4.3 Physical Implementation . . . 65
4.3.1 Circuit Parameters and Memory Characterization . . . 65
4.3.2 Circuit-Level Optimizations . . . 67
4.3.3 Implementation Flow . . . 69
4.3.4 Layout of the Three Implementations . . . 69
4.3.5 Approximate Multipliers . . . 71
4.4 Results and Discussion . . . 71
4.4.1 Reference Model and Functional Validation . . . 72
4.4.2 Power Analysis . . . 73
4.4.3 Power-Decoding Performance Trade-offs . . . 75
4.4.4 Comparison to Existing BCI Accelerators . . . 77

5 AQFE-Based Full BCI System: Challenges and Perspectives 78
5.1 Full-BCI Decoding System . . . 78
5.2 Adaptive-Quality Control Loop . . . 79
5.2.1 Global vs Per-Feature Control . . . 79
5.2.2 Internal Quality Evaluation . . . 80
5.2.3 External Sensing and User Feedback . . . 82
5.2.4 Summary of Closed-loop Feedback Techniques . . . 83

5.2.5 Feasibility of a BCI with Closed-loop Quality Control 84
5.3 Remaining Challenges 85

6 Conclusions **87**
6.1 Summary of Contributions 87
6.2 Discussion and Limitations 88
6.3 Perspectives and Future Work 89
6.4 Concluding Remarks 90

Bibliography **91**

Appendices **111**
A Background on Clinatec BCI Research 111
A.1 Clinical Motivation 111
A.2 BCI Platform 111
A.3 Phase-Amplitude Coupling Study 112
B Porting of Matlab Algorithms to C++ 114
B.1 Decoding Algorithm 114
B.2 Model Update Algorithm 117
B.3 Experimental Measurements 119
C Complex-To-Real (C2R) Post-processing Block 123
C.1 Use Case and Integration 123
C.2 Datapath 123
C.3 Programming Commands 124
C.4 Throughput and Latency 124
C.5 Silicon Area and Power Consumption 124
D Approximate Multipliers 125
D.1 Background on Approximate Multipliers 125
D.2 Algorithmic Evaluation with Behavioral Models 126
D.3 Hardware Exploration and Power Analysis 127

List of Figures

2.1 Simplified BCI Pipeline from Brain Signal Acquisition to Output Commands. 6
2.2 Summary of Linear and Non-Linear Decoders Used in BCI Systems. 10
2.3 Correlation Coefficients for Decoding Motor Parameters Using (A) Low-Frequency Components, (B) Magnitude+Phase, (C) Magnitude-Only, and (D) Phase-Only Features, Adapted from [52]. 13
2.4 Performance of the Brain-Spine Interface during Gait Decoding, Adapted from [88]. 14
2.5 Reference Model for a BCI Decoder Showing a Two-Step Decoding with Feature Extraction. 15
2.6 Block Diagram of a Switch Matrix for Channel Selection. 18
2.7 Block Diagram of a Feature Extraction Block Supporting *Selection* and *Reduction*. 19
2.8 Output Classes per Second as a Function of the Input Data Rate (bits/s). The Numbers Represent the References in Table 2.2. 24
2.9 Scatter Plot of the Information Transfer Rate (bits/min) as a Function of the Classes per Second. 25
2.10 Power Consumption (mW) as a Function of the Input Data Rate (bits/s). Non-Motor Decoding Circuits Are Grouped Together (grey), and Motor Decoding Circuits Are Grouped According to the Signal Type: Green for EEG, Red for ECoG, and Blue for MEA. 27
2.11 Power per Channel (mW) as a Function of the Information Transfer Rate (bits/min). 28

3.1 Examples of Three Common Wavelets: Morlet, Mexican Hat, and Haar. . . 31
3.2 Polynomial Approximations of the Morlet Wavelet. (a) Regression Fits from `np.polyfit`. (b) Interpolation Fits Passing through Extrema. 35
3.3 Spline Approximations of the Morlet Wavelet. (a) Cubic Spline Interpolations with Different Numbers of Knots. (b) B-Spline Least-Squares Fits with Different Numbers of Control Points. 35
3.4 Piecewise Linear Approximation of the Morlet Wavelet. 38
3.5 Piecewise Constant (Quantized) Approximation of the Morlet Wavelet . . . 39
3.6 Overview of the Four CWT Methods: (a) Direct CWT in the Time Domain, (b) FFT-Based CWT in the Frequency Domain, (c) QCWT with a Piecewise Constant Approximation, and (d) PLCWT with a Piecewise Linear Approximation. 40
3.7 Number of Additions as a Function of Buffer Size for Different CWT Methods. 43

3.8 Number of Multiplications as a Function of Buffer Size for Different CWT Methods. 43
3.9 Energy for a Single CWT Operation Using Fixed-Point Arithmetic, as a Function of Buffer Size. 45
3.10 Energy for a Single CWT Operation Using Floating-Point Arithmetic, as a Function of Buffer Size. 45
3.11 Mean Per-Band Energy as a Function of the Number of Extracted Bands F at 100 ms for All CWT Methods 46
3.12 Energy per Signal Channel as a Function of Q (for QCWT) and J (for PLCWT) for the BCI Competition IV Dataset Using 32-Bit Fixed-Point Precision. 49
3.13 Energy per Signal Channel as a Function of Q (for QCWT) and J (for PLCWT) for the 8-DoF Avatar Control Dataset Using 32-Bit Fixed-Point Precision. 49
3.14 Correlation Coefficient for the (a) Real and (b) Imaginary Parts of the Outputs as a Function of Quantization Levels Q for the QCWT. 51
3.15 Correlation Coefficient for the (a) Real and (b) Imaginary Parts of the Outputs as a Function of Subdivision Size J for the PLCWT. 52
3.16 Correlation Coefficient of the L_2-Norm Features as a Function of Total Energy for the (a) QCWT and (b) PLCWT, Before and After Applying the Linear Correction. 52
3.17 Balanced Accuracy Loss vs. CC for the (a) 2-Class and (b) 5-Class BCI Competition IV Tasks. Each Point Corresponds to a Wavelet Approximation. 53
3.18 Balanced Accuracy Loss vs. CC for the 8-DoF Avatar Control Dataset. . . 54
3.19 Confusion Matrices for the BCI Competition 5-class Task (**S3** Dataset) Using FFT Baseline, QCWT-C and PLCWT-C Configurations. 56
3.20 Confusion Matrices for the BCI Competition 5-class Task (**S3** Dataset) Using FFT-CWT, QCWT-CD and PLCWT-CD Configurations. 56
3.21 Distributions of Classification Metrics across Sessions for the 8-DoF Avatar Dataset Using FFT-CWT, QCWT, and PLCWT Configurations. 57

4.1 Overview of a BCI System with Adaptive-Quality PLCWT Feature Extraction 60
4.2 Wavelet Approximation for Different Subdivision Sizes. 61
4.3 Top-Level Block Diagram of the AQFE Showing Compute Units (IntU, SpSMU, SpRU), the CtlU, and Associated Memories 63
4.4 RAM Block Sizes for (a) **AQFE-1**, (b) **AQFE-2**, and (c) **AQFE-4**. . . . 66
4.5 Bytes Accessed per RAM for (a) **AQFE-1**, (b) **AQFE-2**, and (c) **AQFE-4**. 66
4.6 Normalized Energy per Bit Access for (a) Read and (b) Write Accesses as a Function of Depth for Small RAM Macros, Using Different Memory Cut Widths. 67
4.7 Normalized Energy per Bit Access for (a) Read and (b) Write Accesses as a Function of Depth for Large RAM Macros, Using Different Memory Cut Widths. 68
4.8 Leakage Power per Bit as a Function of Depth for Large RAM Macros, Using Different Memory Cut Widths. 68
4.9 Post-Layout View of **AQFE-1** in 22 nm FDSOI. 70
4.10 Post-Layout View of **AQFE-2** in 22 nm FDSOI. 70
4.11 Post-Layout View of **AQFE-4** in 22 nm FDSOI. 71

4.12 Average Correlation at HQ as a Function of the Wavelet Coefficient Bitwidth. 72
4.13 Correlation of the (a) Real, (b) Imaginary Part, and (c) Module of the PLCWT Complex Components across Different Frequencies and Approximation Levels. 73
4.14 Leakage and Total Power Profiles for the (a) **AQFE-1**, (b) **AQFE-2**, and (c) **AQFE-4** Implementations. 74
4.15 Zoomed Leakage and Power Profile for the **AQFE-4** at **MQ**. 74
4.16 Average Power vs. Subdivision Size J for (a) **AQFE-1**, (b) **AQFE-2**, and (c) **AQFE-4**. 75
4.17 Power Composition Across Lane Configurations for Different Quality Parameters. 76

5.1 Closed-Loop BCI Decoding with Runtime Quality Control Using the AQFE. 79
5.2 Quality Adjustment Using (a) Global and (b) Per-Feature Control. 80
5.3 Experimental Setup with an EEG Headset and EMG Sensors, Adapted from [177]. 83
5.4 Controller Power P_{ctl} as a Function of the Duty Cycle d. 85

A.1 Restoring Mobility After a Spinal Cord Injury Using an ECoG-Based Battery-Powered BCI and a Spinal Stimulator. 112
B.1 ECoG Decoding Flow Showing the Steps from Signal Acquisition to Output Decisions 114
B.2 Compute Time for Decoding in Scenario S7680, Running on an x86 Host. . 116
B.3 Amplitude of the Real and Imaginary Parts of Morlet Wavelet FFT Coefficients. 117
B.4 High-Level Flow of the REW-NPLS Model Update Algorithm (15 s Update). 117
B.5 Distribution of Compute Time for the Model Update Algorithm. 118
B.6 Acceleration of the Update Algorithm (Scenario S7680, A72 Platform). . . 119
B.7 Execution Time on Two Embedded Platforms (A72 and A76) for (a) Inference and (b) Model Update. 120
B.8 Power Consumption vs. Time for Both Platforms. 121
C.1 Datapath of the C2R Module with Selectable Norms and Optional Scaling. 123
D.1 Example of a BAM with Horizontal and Vertical Cuts, Adapted from [65]. 125
D.2 Average CC vs. Vertical Cut Parameter v for BAM Multipliers with $h = 0$. 126

List of Tables

2.1 Decoding Algorithms Tested for Continuous Motor Parameter Estimation [52]. 12
2.2 Summary of State-of-the-Art Brain Signal Decoding Hardware Implementations . . . 16
2.3 Summary of BCI Circuit State-of-the-Art Optimizations . . . 23

3.1 Number of Operations for Existing and Proposed CWT Methods, with and without Exploiting Optimizations such as Wavelet Symmetry or Sparsity. . 42
3.2 Energy per Operation at 45 nm for 32-Bit Precision, from [60]. . . . 42
3.3 BCI Competition IV, 2-Class Task Classification Metrics (in %). . . . 55
3.4 BCI Competition IV, 5-Class Task Classification Metrics (in %). . . . 55
3.5 Avatar 8-DoF: Classification Metrics (Mean across Sessions, in %). . . . 57
3.6 Summary of the Balanced Accuracy Loss and Energy Reduction Factor of the Approximate Methods Compared to the FFT-CWT Baseline Accuracy and Energy Cost. . . . 58

4.1 RAM Sizes for Different AQFE Configurations . . . 64
4.2 RAM Read and Write Accesses per Inference in the AQFE Module . . . 64
4.3 Summary of SoA Brain Signal Decoding Hardware Implementations . . . 77

5.1 Summary of Closed-Loop Feedback by Source and Control Scope. . . . 84

B.1 Characteristics of Use Scenarios . . . 115
B.2 Compute Capacity of Two Embedded Platforms. . . . 120
B.3 Comparison of Cortex A72 and A76 SoCs for Scenario *S7680*. . . . 121
B.4 Power and Energy Comparison between A72 and A76 SoCs for Scenario *S7680*. . . . 122
C.1 C2R External Interface Parameters . . . 124

List of Acronyms

ADC Analog-to-Digital Converter.
AQFE Adaptive Quality Feature Extractor.

BAM Broken-Array Multiplier.
BB Base-Band.
BCI Brain-Computer Interface.
BSI Brain Spine Interface.

C2R Complex-to-Real.
CC Correlation Coefficient.
CNN Convolutional Neural Network.
CpS Classes per Second.
CtlU Control Unit.
CWT Continuous Wavelet Transform.

DCT Discrete Cosine Transform.
DNC Distinctive Neural Code.
DoF Degree of Freedom.
DR Decision Rate.
DSP Digital Signal Processor.
DT Decision Tree.
DVFS Dynamic Voltage/Frequency Scaling.
DWT Discrete Wavelet Transform.

EB Energy Band.
ECoG Electrocorticography.
EEG Electroencephalography.
EMG Electromyography.
EOG Electrooculography.
ERP Event-Related Potential.

FB Full-Band.
FD Fourier Descriptor.
FE Feature Extractor.

FEP Feature Evaluation Parameter.
FFT Fast Fourier Transform.
fNIRS functional Near-Infrared Spectroscopy.
FPGA Field Programmable Gate Array.

HC Hjorth Complexity.
HFO High-frequency Oscillations.
HMM Hidden Markov Model.
HP Hjorth Parameter.

IDR Input Data Rate.
iFFT Inverse Fast Fourier Transform.
IntU Integrator Unit.
ITR Information Transfer Rate.

KNN K-Nearest Neighbor.

LDA Linear Discriminant Analysis.
LFC Low Frequency Component.
LL Line Length.
LMP Local Motor Potential.
LR Linear Regression.
LSTM Long Short-Term Memory.
LUT Lookup Table.

MCU Microcontroller Unit.
MEA Microelectrode Array.
MEG Magnetoencephalography.
MFR Mean Frequency.
MLP Multilayer Perceptron.
MSLM Markov Switching Linear Model.
MST Modified Stockwell Transform.

OLE Optimal Linear Estimator.

PAC Phase-Amplitude Coupling.
PCA Principal Component Analysis.
PD Parkinson Disease.
PFD Petrosian Fractal Dimension.
PLCWT Piecewise Linear Continuous Wavelet Transform.
PLV Phase Locking Value.
PpC Power per Channel.
PSD Power Spectral Density.

QCWT Quantized Continuous Wavelet Transform.

RAM Random Access Memory.

REW-NPLS Recursive Exponentially Weighted N-way Partial Least Squares.
RF Random Forest.
RNN Recurrent Neural Networks.
RWE Relative Wavelet Energy.

SBP Spiking Band Power.
SIMD Single Instruction Multiple Data.
SNN Spiking Neural Network.
SoA State-of-the-Art.
SoC System-on-Chip.
SpRU Sparse Reduction Unit.
SpSMU Sparse Shift Multiplication Unit.
SR Sampling Rate.
SRAM Static Random Access Memory.
SSKF Steady-State Kalman Filter.
SSVEP Steady-State Visually Evoked Potentials.
STFT Short Time Fourier Transform.
SVM Support Vector Machine.

TRCA Task-Related Component Analysis.

WT Wavelet Transform.

Chapter 1

Introduction

According to the World Health Organisation there are 15 million people globally who suffer from spinal cord injury [4]. For many of these individuals, Brain-Computer Interfaces (BCIs) offer a way to regain function by translating neural activity into control commands that drive assistive devices and rehabilitation systems [37, 94]. Therefore, making such systems widely accessible for everyday use has the potential to improve the lives of millions of people.

While this work focuses on motor applications, BCI systems can also be used for speech decoding [22, 58, 100], emotion recognition [61], or detection of visual and auditory stimuli [19]. More generally, across these applications, the main challenge is to decode brain signals in real-time, under tight power and area budgets, especially for portable and implantable systems [124].

Over the last two decades, advances in recording hardware, signal processing, and decoding algorithms have made BCIs more accurate, faster, and closer to real-world use [70]. In line with these developments, the goal of this work was to reduce the power for BCI decoders, while preserving high decoding accuracy with low latency. This thesis was conducted at CEA-List, which has developed ultra-lower power embedded systems, in collaboration with Clinatec (CEA-Leti), where research on BCIs for motor recovery and rehabilitation has been underway the last two decades.

A typical BCI pipeline follows three steps: (i) acquisition and preprocessing of the input data, (ii) computing intermediate representations, also called features, from the input signals, and (iii) decoding that consists of feeding the features to a classifier or regressor to generate the outputs. Feature extraction is thus an intermediate step that turns the raw data into discernible features that a downstream model can decode.

A large variety of features have been explored [162], depending on the applications, and feature engineering continues to evolve. Among these, time-frequency features are often used, since neural rhythms consist of oscillations at specific frequencies that evolve over time. In this thesis, we focus on this type of feature, extracted using the Continuous Wavelet Transform (CWT), which has been widely used for previous BCI applications [87, 88]. In practice, the CWT is computed by convolving the signal with scaled versions of a mother wavelet across many frequency bands and times. This can be done with filters in the time domain or with a Fast Fourier Transform (FFT) method, where the convolution becomes a multiplication in the Fourier domain. Existing CWT implementations are computationally intensive and can become a bottleneck for power-constrained BCI systems.

At Clinatec, a BCI system based on CWT features is used with real participants in

online experiments for motor decoding. A typical session starts with collecting labeled data, followed by supervised training of the decoder, then the participant uses the system in inference mode. Since the signals vary across tasks and drift over time, the model is periodically updated to keep performance stable.

I started the thesis by becoming familiar with BCIs and then optimizing the existing software at Clinatec, with the help of my supervisors and an intern student. This work made it possible to perform the decoding and the model update algorithms on a small, embedded processor, resulting in a portable platform consuming 10× less power than the previous implementation on a portable PC. The embedded implementation received the jury prize at the "Assises de l'Embarqué 2025", in Paris [2]. These optimizations have been published in the 21st International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT 2025) [137] and in the Design, Automation and Test in Europe (DATE) 2025 Conference [45].

As the goal of this thesis was to sufficiently lower the power consumption for the decoding step so that this could be performed, in the long term, directly in the implant, I profiled the decoding algorithm and benchmarked existing BCI systems with comparative metrics that I proposed for a fair comparison. The background study and the proposed metrics were published in 2025, in the journal Frontiers in Human Neuroscience [134]. The findings showed that the main compute bottleneck is feature extraction, which, in our system, is dominated by the CWT. I therefore developed two approximate alternatives for the CWT, which significantly reduce the compute complexity of feature extraction. I then implemented one of the proposed methods as a hardware block, evaluated its performance, and proposed a way of integrating it with a post-processing module in a full BCI.

At the algorithmic level, the two CWT variants that we propose are based on different ways of approximating the wavelets. The Quantized Continuous Wavelet Transform (QCWT) approach is based on a piecewise-constant approximation to reduce the number of multiplications required for the convolution. The alternative Piecewise Linear Continuous Wavelet Transform (PLCWT) method preserves continuity of the approximated wavelets, while requiring fewer arithmetic operations. For both methods, I analyzed the correlation between approximate and exact features, and introduced a linear output correction that partially compensates approximation errors when extracting energy using non-linear norm operators. These approximations were shown to maintain the discriminative information required by a BCI decoder, while significantly reducing computation. In addition, I evaluated the proposed methods on representative BCI datasets for motor tasks, and estimated energy savings to be up to 4×, based on the number of arithmetic operations. I presented the algorithmic contributions, at the 13th International Winter Conference on Brain-Computer Interface 2025 [135].

To demonstrate that the proposed algorithms could be efficiently implemented in hardware and to quantify the energy savings, a wavelet-based Adaptive Quality Feature Extractor (AQFE) was designed to physically implement the PLCWT efficiently, in a 22 nm FDSOI technology. The design maps the algorithmic approximations to specialized compute units. Post-layout analyses quantify the area and power consumption of the designed module, showing that it has a 2× lower area than existing circuits, while consuming at least 4× less power than prior digital feature extractors. This work was accepted as a regular paper in the Design, Automation and Test in Europe Conference 2026.

A key idea in this thesis is to dynamically adjust the quality of the features according to the task complexity. In the AQFE, a tunable parameter controls the approximation

level. Higher quality preserves more detail, while lower quality reduces energy. This lets us select the trade-off between quality and energy cost of feature extraction. In contrast to existing approaches such as Neuraltree [149], which adapt the decision process by selecting a subset of features, our method controls the quality of the extracted features, meaning that the two strategies can be combined. The concept of dynamically varying the quality of the features based on a closed-loop feedback was patented by the CEA (patent FR2501705).

Beyond the main objective of the thesis, I also made contributions to related topics. In collaboration with two researchers at Clinatec, we studied the use of Phase-Amplitude Coupling (PAC) [165] features for motor decoding in long-term Electrocorticography (ECoG) recordings. PAC quantifies how the phase of a low-frequency component in the brain signals modulates the amplitude of a higher-frequency band. In this collaboration, I discussed and refined the methodology, wrote scripts to test the PAC features on the existing database offline and in a pseudo-online setup, and contributed to writing a paper that was published in the journal Frontiers in Human Neuroscience [95].

In addition, I contributed to a circuit called BCI-DSPI1, developed for the NEMO-BMI project. This circuit computes the CWT with traditional FFT-based techniques. My contribution was the RTL design of a post-processing Complex-to-Real (C2R) hardware module that computes per-band energy from the CWT outputs using selectable norms, as detailed in Appendix C. The block can aggregate energy over custom windows while applying optional normalization. It was integrated into a full decoding circuit. I also contributed to the verification flow: unit tests of the FFT module, and algorithmic verification of the top module to ensure functional equivalence with the reference software. The circuit was taped out in October 2025 and is currently being fabricated.

Beyond the scientific contributions, I supervised an internship that supported the embedded software ramp-up and optimization. In parallel, I presented communications at international and local conferences, reported progress at NEMO-BMI project meetings, and gave internal presentations at the CEA (PhD forums, student and lab meetings). I also had the opportunity to present parts of this work at MT180 (Ma thèse en 180 secondes), a three-minute public outreach event where PhD candidates explain their research to the general public. I was selected at the Grenoble Academy level to present at the Rhône-Alpes regional level and the video can be see here https://youtu.be/8mN_39qcOYY?si=wlFsV4rc1Y9x05BQ&t=4207.

This manuscript is organized as follows. Chapter 2 reviews brain signals, existing feature extraction methods, and frequently used BCI decoders. In addition, existing hardware implementations are analyzed using proposed metrics to enable fair system comparison. Chapter 3 introduces the two proposed approximate CWT methods, and presents their performance in terms of correlation with the exact CWT and their decoding accuracy, along with the estimated energy savings. Chapter 4 describes an adaptive hardware implementation for wavelet-based feature extraction, including post-layout characterization, and a comparison to existing BCI accelerators. Chapter 5 discusses the integration of the extractor into a full system and the remaining challenges for deployment. We finally conclude in Chapter 6 and outline perspectives for future work.

Additional material is provided in the appendices. Appendix A describes Clinatec's existing work on BCIs and the PAC study in which I participated. Appendix B details the embedded software optimization for the decoding and the model update algorithms, and presents benchmark results. Appendix C details the hardware development of the C2R block, with the RTL interface, synthesis and post-layout area and power. We present, in

Appendix D, the exploration of approximate Broken-Array Multipliers (BAMs), and give preliminary results and details on the applied methodology, a path that was explored but which turned out not to bring significant benefits in our application.

It is my hope that the results of this thesis will help bring reliable, low-power BCIs closer to everyday motor rehabilitation.

Chapter 2

Background and State of the Art

Brain-Computer Interfaces (BCIs) enable the translation of neural activity into speech, for motor control, or for rehabilitation. Depending on the brain signal characteristics and application domains, BCI systems are based on different types of algorithms. The typical approach for processing brain signals consists of three main steps: signal acquisition, feature extraction, and decoding. These approaches can vary in complexity, and thus computational cost. The choice of decoding algorithm and feature representation significantly influences system accuracy, latency, and energy consumption, which are particularly important in embedded and implantable devices. To support real-time low-power operation, many BCI systems rely on dedicated hardware. As the field is still in its early stages, there are for standardized metrics for comparing BCI system, especially when considering hardware-software trade-offs in system design. This chapter reviews the current landscape of BCI systems, and is organized as follows:

- Section 2.1 summarizes the algorithms used in different BCI applications, including techniques for feature extraction and decoding methods.
- Section 2.2 reviews hardware implementations of such systems, highlighting optimizations at the algorithmic, architectural, and circuit levels.
- Section 2.3 presents a set of proposed metrics that can be used to better compare systems, and also discusses key limitations of current approaches, which motivates the problem addressed in this thesis.

The last two sections of this chapter are based on a previous literature review that was published in [134].

2.1 Overview of Feature Extraction and Decoding Algorithms in BCI Systems

Different types of neural signals can be recorded using invasive or non-invasive methods, offering different trade-offs in spatio-temporal resolution. BCIs rely on a multi-stage processing pipeline to convert the recorded neural activity into commands or outputs. Two of the most critical stages in this pipeline are *feature extraction*, which transforms neural signals into informative representations, and *decoding*, which maps these features to control actions, as shown in Figure 2.1.

Depending on the signal type and application requirements, various feature representations and decoding algorithms have been proposed. These range from handcrafted

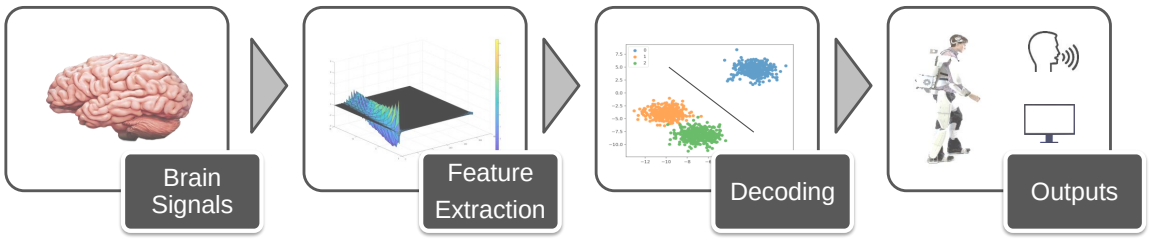


Figure 2.1: Simplified BCI Pipeline from Brain Signal Acquisition to Output Commands.

frequency-based features combined with linear classifiers [88, 155] to deep learning approaches trained end-to-end [40, 76, 128]. This section reviews the most commonly used techniques in both steps.

2.1.1 Brain Signals for BCI Systems

Methods for brain signal recording can range from fully-invasive to non-invasive. Microelectrode Arrays (MEAs), implanted directly into the brain tissue, offer a high spatial resolution as they are able to capture single-neuron signals [97, 111, 122] while Electrocorticography (ECoG) arrays, placed on the surface of the brain, measure signals averaged over thousands of neurons with limited invasiveness [96, 99, 150]. Non-invasive techniques can also be used for brain signal acquisition. Electroencephalography (EEG) signals detect electric signals averaged over a larger number of neurons than ECoG [150] and can be used for brain signal decoding [30, 156, 175, 182].

Other non-invasive methods include Magnetoencephalography (MEG) [170] and functional Near-Infrared Spectroscopy (fNIRS) [114]. MEG measures magnetic activity in neurons, using machines that are physically large, hence not portable. The fNIRS method is based on detecting changes in hemodynamic activity by measuring variations in oxyhemoglobin and deoxyhemoglobin concentrations. Although fNIRS devices have become more portable, they do not meet the needs for real-time motor assistive applications, as they have a response-time of a few seconds [119], much slower than methods based on electrical signals.

2.1.2 State-of-the-Art Feature Extraction Methods

The extraction of informative features is often a crucial step in a BCI system. Depending on the task and the type of neural signals, several types of features have been explored. Features aim to maximize separability between classes while generally extracting intrinsic properties of the input signals. The choice of features not only influences decoding performance but also determines the computational complexity and energy requirements of the BCI system.

2.1.2.1 Time-Frequency and Spatial Features

Frequency-domain features, such as Power Spectral Density (PSD), are widely used to capture neural rhythms associated with motor, visual, or speech-related activity. These features are typically extracted via analog [93, 189] or digital [58, 88, 141, 176, 188] bandpass filtering, which is often followed by post-processing steps such as z-score [148, 176, 188], energy computation [141], or smoothing [148, 188]. High-γ activity (75-200 Hz), for

example, is commonly associated with motor intention [176], while lower-frequency bands like μ(8-12 Hz) and β(13-30 Hz) reflect desynchronization during movement [98].

Time-frequency features extend this concept by capturing changes in power across the time and frequency domains. The wavelet transform [14, 25, 28, 79, 87, 140] is employed for this purpose, particularly in applications involving upper and lower limb movement [20, 88] or speech decoding [58, 100]. Other methods such as the Modified Stockwell Transform (MST) [158] can also be used to extract time-frequency features for BCIs [63] and can even be extended to other bio-signals [159].

Spatial localization of the brain electrodes also carries relevant information. Due to the size and placement of the electrodes, the recorded signals represent spatial averages over a chosen population of neurons. Recording from a large number of electrodes can reveal the brain's spatial organization, either by covering a broad cortical surface [96, 144, 149] or by capturing variability across larger depth [111, 187]. Techniques such as inter-electrode correlation [92, 98] have also been explored to extract this spatial information. These spatial patterns can vary across tasks and frequency bands, offering an additional layer of information that can be used for decoding.

2.1.2.2 Advanced and Nonlinear Features

Beyond spectral, temporal and spatial information, phase-based features have shown potential in improving decoding accuracy, especially when combined with spectral features. For example, [92] demonstrated classification accuracy improvements using phase features comparable to power-based features, while [52] reported significantly higher velocity decoding correlations using phase information. Moreover, Phase-Amplitude Coupling (PAC) features, which capture the modulation of high-frequency amplitude by the phase of lower-frequency components [165], have been investigated in [95] to improve motor imagery decoding in BCI applications. They have also been used in [80] to dynamically identify optimal frequency bands, improving the decoding performance.

Furthermore, other characteristics have been explored for BCIs. For example, Higuchi fractal dimension [6, 8] and Hurst exponent [9] aim to capture regularity in EEG signals. Moreover, entropy features, including Renyi [71], permutation [116] and Shannon entropy [120] have been shown to discriminate motor states as they reflect the variability of brain signals. Other parameters such as Lempel-Ziv complexity [85] or Hjorth parameters [149, 169] capture neural signal nonlinear dynamics. These features often complement traditional spectral or time-frequency features to improve decoding performance.

Conclusion 2.1.2

In the literature on BCIs, a wide range of features have been investigated. Among these, spectro temporal features remain those that are most widely used.

2.1.3 State-of-the-Art BCI Decoders

Decoding algorithms transform extracted features into outputs such as movement or speech. Over the years, a wide variety of decoding approaches have been proposed, ranging from simple linear models with low computational cost to sophisticated deep learning architectures to exploit more complex brain dynamics. These decoders differ in their adaptability and computational requirements, which makes their selection dependent on the intended application and hardware constraints. This section reviews the main

decoders employed in BCIs, and discusses their strengths and limitations.

2.1.3.1 Linear Decoders

Linear decoders are widely used for BCI systems. They assume a linear mapping between extracted features and outputs. Their main advantage lies in their computational efficiency and simple interpretation, making them particularly suitable for real-time and embedded BCI implementations. Such decoding models have shown good performance across a wide range of tasks, from motor imagery and Steady-State Visually Evoked Potentials (SSVEP) to Event-Related Potential (ERP). Among these, several variants can be distinguished, including linear classifiers and regression models for continuous decoding.

- **Basic linear classifiers** aim to find a hyperplane boundary in the feature space to separate patterns from different classes. In BCIs, the most common linear classifiers are Linear Discriminant Analysis (LDA), logistic regression, and Support Vector Machines (SVMs). LDA assumes that all classes follow the same Gaussian distribution [17], and it has often been used in BCI for motor decoding [28, 67], stimulus detection [59], and speech generation [144]. In contrast, logistic regression models the probability of each class directly, and fits a linear boundary via maximum likelihood. It has been used for epileptic seizure detection [36], in addition to motor [72] and ERP [101] decoding. Similarly, a linear SVM tries to find the hyperplane that maximizes the margin between classes in the feature space, but unlike LDA or logistic regression, it focuses on the samples closest to the decision boundary, which are called the support vectors. SVM decoders have been widely used for epileptic seizure detection [178, 186] and, in BCIs, for motor decoding [155]. Variants such as hierarchical SVM [41] and ensemble SVM [81], based on the fusion of several classifiers, have also been explored to decode brain signals.
- **Regression-based continuous decoders** directly map brain signals to continuous control outputs such as hand trajectories, joint angles, or cursor velocities. Unlike classifiers, regression models provide continuous values as predictions to enable smooth control in motor BCI tasks. The Linear Regression (LR) is an approach that uses weighted sums of neural signals or features to estimate the output. An Optimal Linear Estimator (OLE) has been used in [176] to decode upper limb movements. Other LR methods have been applied in both intracortical and ECoG-based BCIs for multidimensional cursor and limb control [145, 163]. Beyond linear regression models, linear state-space models such as the Kalman filter take into account temporal dynamics by predicting the next movement according to the current one and the observed neural activity. This smoothing makes them well-suited for continuous task decoding such as wrist movement [129]. Variants such as the unscented Kalman filter have been explored to better capture nonlinear dynamics [83]. Furthermore, regression methods have also been extended with spatial filtering to improve the signal-to-noise ratio of EEG features. For instance, spatial filters tailored for regression tasks have been proposed for better decoding accuracy in motor imagery experiments [181]. Finally, linear regression models can be mixed with classification models such as in [20], where a Markov Switching Linear Model (MSLM) model outputs a classification that is then used to select an "expert", consisting of a linear model, which outputs the estimated kinematic parameter.

Conclusion 2.1.3.1

Linear decoders are among the most established approaches in BCI. They have been applied successfully across many decoding tasks where real-time performance and limited hardware resources are the main constraints.

2.1.3.2 Non-linear Decoders

Non-linear decoders do not require input signals or features to be linearly separable, which allows them to capture more complex relationships between neural activity and outputs. While they generally require more compute power than linear decoders, they often achieve higher accuracy when brain dynamics are non-linear. These methods have been applied to a wide range of BCI tasks, including motor control, and speech decoding, in addition to epileptic seizure detection.

- **Kernel-based classifiers** use non-linear decision boundaries to map the inputs to the outputs. They have been used for EEG motor imagery decoding [31], where different kernels (linear, polynomial, radial basis and sigmoidal) for SVMs were benchmarked against a Multilayer Perceptron (MLP) decoder. Among non-linear kernels, radial basis functions are commonly used for motor BCIs such as left and right hand movement classification [51], lower-limb imagery [56] and epileptic seizure detection [82]. Similarly, Gaussian process classifiers [172] model brain signals as distributions, offering uncertainty estimates. Such classifiers have been applied in motor imagery and SSVEP decoding tasks.
- **Tree-based methods** such as decision tree and random forest methods capture non-linear interactions between features. A Decision Tree (DT) partitions the feature space into regions based on a depth-dependent condition. The process continues recursively until a decision criterion is reached, which corresponds to the output of the tree. A Random Forest (RF) is based on combining many DTs to reduce the risk of overfitting for high dimensional spaces. For classification, the forest predicts the class chosen by the majority of trees. For regression, it outputs the average prediction of the trees. These decoders have been used in motor BCIs to decode a 2D cursor movement using EEG signals [15]. In addition, they have been used in other paradigms such as word typing using ERP [7], SSVEP classification [183] and seizure detection [49]. Hardware solutions such as NeuralTree [149] have been developed to implement tree-based decoders for ECoG and EEG signals, highlighting their suitability for implantable low-power applications.
- **Neural networks** implement non-linear functions by stacking layers of interconnected neurons with non-linear activation functions. Early neural networks were applied to classification paradigms such as grasp decoding [123]. With the advances in deep learning methods, more complex architectures such as Convolutional Neural Networks (CNNs) [40, 126] and Recurrent Neural Networkss (RNNs) [76, 128, 166] have been used for BCI applications, especially in end-to-end decoding, offering improvements in accuracy, but requiring larger datasets and more compute requirements to train such models. A CNN-based architecture achieved a mean accuracy of 70.6% on the BCI Competition IV-2a dataset, compared to 67.0% for SVM (p-value$= 0.0091$) [139]. In [18], a Transformer-based model (TransformerNet) reached an average accuracy of 85.97%, largely outperforming linear methods and other compact CNNs such as EEGNet. Hybrid techniques, combining several decoding models have also been shown to increase the decoding performance. For example, in [146],

the authors explore a hybrid method combining a CNN and a Long Short-Term Memory (LSTM) model. In [168], a fusion of five different architectures is used to improve classification accuracy. Compact variants such as EEGNet [78] have also been developed to enable deployment on embedded hardware. Furthermore, Spiking Neural Networks (SNNs), a variant based on spike representation of signals, demonstrated promising results when implemented on neuromorphic hardware [23].

- **Other non-linear methods** exist and have been explored for brain signal decoding. Among these, K-Nearest Neighbor (KNN) [32] and Task-Related Component Analysis (TRCA) [182] have been explored in BCIs. In addition, probabilistic models such as Naïve Bayes classifiers [91, 152] provide simple and interpretable decoders. Moreover, Markovian models like Hidden Markov Models (HMMs) have been applied for BCI decoding [20, 57], due to their ability to represent state transitions over time. These methods remain important for tasks requiring real-time robustness.

Conclusion 2.1.3.2

Non-linear decoders have shown performance gains in tasks such as motor imagery and speech decoding. These come from their ability to model complex non-linear brain dynamics. However, their reliance on kernel operations or multi-layer processing increases computational demands compared to simple linear models.

2.1.3.3 Summary of BCI Decoders

We reviewed a wide range of linear and non-linear decoding methods. To provide a more compact overview, Figure 2.2 summarizes the main families of decoders used in BCIs and gives examples within each category.

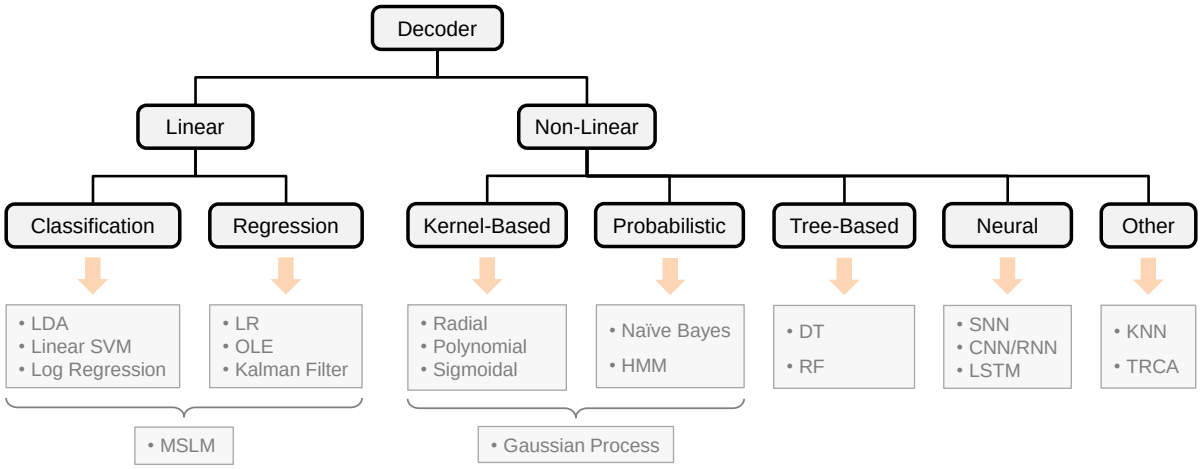


Figure 2.2: Summary of Linear and Non-Linear Decoders Used in BCI Systems.

2.1.4 Examples of Feature Extraction and Decoding Systems

Over the years, research in BCI has driven the development of feature extraction and, more generally, brain signal decoding techniques. Although not exhaustive, this section presents key BCI system examples, selected to emphasize the diversity of applications and methods.

2.1.4.1 Decoding Object Categories from Temporal Correlations of ECoG Signals

In [92], visual object classification from ECoG signals was investigated by comparing different types of features. Subjects were presented with 120 images from 24 categories while ECoG activity was recorded. Feature sets included band-power (1-10, 10-20, 20-30, 30-80 and 80-150 Hz), inter-electrode correlations, and phase information at representative frequencies, which were then classified using a linear SVM.

The study demonstrated that correlation-based features significantly outperformed both power and phase features, and that combining correlation with band-power further improved accuracy. Interestingly, the analysis showed that low-frequency bands (1-10 Hz) yielded the highest decoding performance, while shuffling experiments revealed that high-frequency correlations mainly reflected relative timing between electrodes rather than absolute latency. Moreover, correlation features provided faster decoding, reaching above-chance accuracy within 25 ms compared to 100 ms for power features.

This work highlights the potential of inter-electrode correlation features for BCI decoding, suggesting that spatio-temporal dependencies across electrodes may carry more information than spectral features alone. However, the computational cost of extracting pairwise correlations (scaling quadratically with the number of electrodes) raises challenges for real-time or embedded implementations.

2.1.4.2 Phase-amplitude coupling-based adaptive filters for neural signal decoding

The work in [80] proposed a decoding strategy that leverages the physiological phenomenon of PAC, where the phase of low-frequency components modulates the amplitude of high-frequency activity. The authors introduced PACNet, a deep-learning based decoder in which adaptive band-pass filters are tuned to extract high-gamma sub-bands most correlated with low-frequency rhythms.

Using ECoG datasets for finger movement [104] and a visual discrimination task with two image categories (face, house) shown with 400 ms pause intervals. In this experiment, the PAC method is used to choose optimal frequency bands for band-pass filters to extract features before applying a neural network for classification. This adaptive filtering yielded small but statistically significant improvements compared to using the entire high-gamma band, achieving decoding accuracies of 62.5% for finger flexion and 97.8% for visual discrimination.

While the results show that PAC-based adaptive filtering can increase accuracy, the approach adds computational complexity to the system. Nevertheless, the study highlights the potential of coupling phase and spectral information to improve ECoG decoding and reduce latency in future BCI systems.

2.1.4.3 ECoG-based Brain-Computer Interface Using Relative Wavelet Energy and Probabilistic Neural Network

The authors in [192] addressed a two-class decoding task from the BCI Competition III dataset, distinguishing between imagined finger and tongue movements using ECoG recordings. The approach relied on computing relative wavelet energy features using Discrete Wavelet Transform (DWT) coefficients across five frequency bands (δ, θ, α, β, γ). A subset of electrodes was selected, yielding 40 features which were then reduced by

applying Principal Component Analysis (PCA) before classification with a probabilistic neural network.

The system achieved an average decoding accuracy of 91.8% (with 99.6% for training and 84% for testing), suggesting strong potential for real-time BCI use. The choice of the DWT allowed for efficient $\mathcal{O}(N)$ feature extraction compared to continuous wavelet transforms, while the probabilistic neural network decoder offered faster training than conventional backpropagation neural networks. Together, these design choices emphasized computational efficiency alongside accuracy.

Although the results were promising, the methodology left some ambiguities, particularly in electrode selection and the mixing of training and test metrics. Nonetheless, this work illustrates how wavelet-based features combined with lightweight classifiers can provide accurate and efficient decoding for practical BCI applications.

2.1.4.4 The role of ECoG magnitude and phase in decoding position, velocity, and acceleration during continuous motor behavior

The role of magnitude and phase features in ECoG-based decoding has been explored in the context of continuous motor tasks, where participants controlled a virtual car by steering [52]. A multiple linear regression model was tested with several input representations: Low Frequency Component (LFC) of the time-domain signal, spectral power features, and Short Time Fourier Transform (STFT) phase and magnitude features, to estimate position, velocity and acceleration. In a first approach, the outputs are decoded in the time domain and correspond to the low-pass filtered (at 4 Hz) time values of the kinetic parameters. In a second approach, the model predicts the outputs in the frequency domain, called Fourier Descriptors (FDs), and the kinetic parameters are reconstructed by applying an inverse STFT. The authors test different combinations of inputs and outputs and the resulting decoding algorithms are shown in Table 2.1.

Table 2.1: Decoding Algorithms Tested for Continuous Motor Parameter Estimation [52].

Algorithm	Inputs	Outputs
Alg1	LFC in time domain	Low-pass filtered (4 Hz) in time domain
Alg2	Spectral power band	Low-pass filtered (4 Hz) in time domain
Alg3	Phase + Magnitude of STFT	FD outputs, then inverse STFT
Alg4	Phase of STFT	FD outputs, then inverse STFT
Alg5	Magnitude of STFT	FD outputs, then inverse STFT

The analysis showed that position and velocity could be more reliably decoded than acceleration. Phase features outperformed magnitude features, and their combination yielded significantly higher correlations than LFC alone. Moreover, phase features provided a smoother temporal profile around movement onset. Compared to band-power features, phase information achieved superior performance. The resulting decoding performance for the different feature sets is illustrated in Figure 2.3, where the Correlation Coefficient (CC) between the real and decoded movement parameters is shown as a function of the time offset τ.

These findings suggest that phase carries complementary information to magnitude and LFC features, making it a valuable resource for continuous motor decoding in BCI.

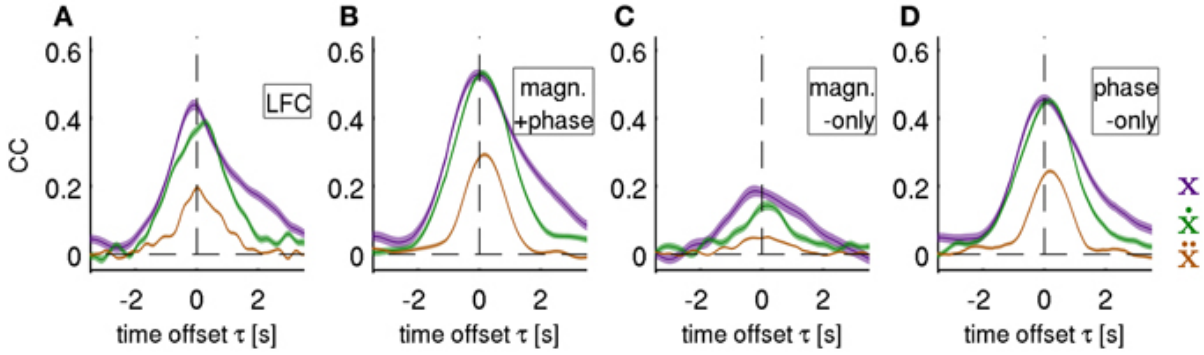


Figure 2.3: Correlation Coefficients for Decoding Motor Parameters Using (A) Low-Frequency Components, (B) Magnitude+Phase, (C) Magnitude-Only, and (D) Phase-Only Features, Adapted from [52].

By reconstructing motor parameters in the frequency domain and applying an inverse transform, this approach offers a path toward more accurate continuous BCI decoding.

2.1.4.5 Feature Engineering for an Efficient Motor-Related ECoG BCI System

The study in [63] investigated eight different feature categories derived from ECoG signals to balance accuracy and latency in real-time BCIs. To assess the optimal time interval for each feature type, the authors introduced a signal split parameter, representing the number of segments over which features are extracted. This allowed them to distinguish between short-range features (better at fine temporal resolution) and long-range features (more stable over longer intervals).

The approach consisted of preprocessing raw ECoG data, computing all feature types across multiple split lengths, and applying a genetic algorithm to select the most relevant subset, which was then classified using an SVM. Experiments on the BCI Competition III dataset [21] and the Motor Basic dataset [103] categorized the features as follows:

- Long-range features: Modified Stockwell Transform (MST), Mean Frequency (MFR), Frequency at Max Power (FMP).
- Mid-range features: Relative Wavelet Energy (RWE), Petrosian Fractal Dimension (PFD).
- Short-range features: Power Spectral Density (PSD), Hjorth Complexity (HC).

On the BCI Competition III dataset, the optimized feature set reached a 93% accuracy and an Information Transfer Rate (ITR) of 20.5 bits/min, outperforming previously reported benchmarks (17.1 bits/min in [185]). The genetic algorithm revealed that optimal subsets at different feature counts were not strict supersets of one another, highlighting that feature utility depends on the dimension budget. This work illustrates how careful feature engineering can improve accuracy while reducing latency in BCIs.

2.1.4.6 Walking naturally after spinal cord injury using a brain-spine interface

In [88], the authors demonstrated a Brain Spine Interface (BSI) that enabled a paraplegic patient to walk naturally by combining brain activity decoding with spinal cord stimulation. ECoG signals were recorded from two 64-electrode implants [99], processed with

a wavelet transform to extract spectral features. These were fed into a MSLM combined with a HMM for state classification.

Two decoding schemes were implemented: a seven-state model controlling six joints plus an idle state, and a simplified three-state model (left hip, right hip, idle) optimized for walking. The models were trained with the Recursive Exponentially Weighted N-way Partial Least Squares (REW-NPLS) algorithm, enabling incremental learning with relatively short calibration sessions. Closed-loop performance achieved less than 1.1 s end-to-end latency, with reported decoding accuracies of 74% for joint control and 83% for muscle group control in the seven-state mode, and 97% for the three-state walking decoder. The performance remained stable up to 11 months after the initial surgery. The experimental results showing the decoding and walking performance are shown in Figure 2.4.

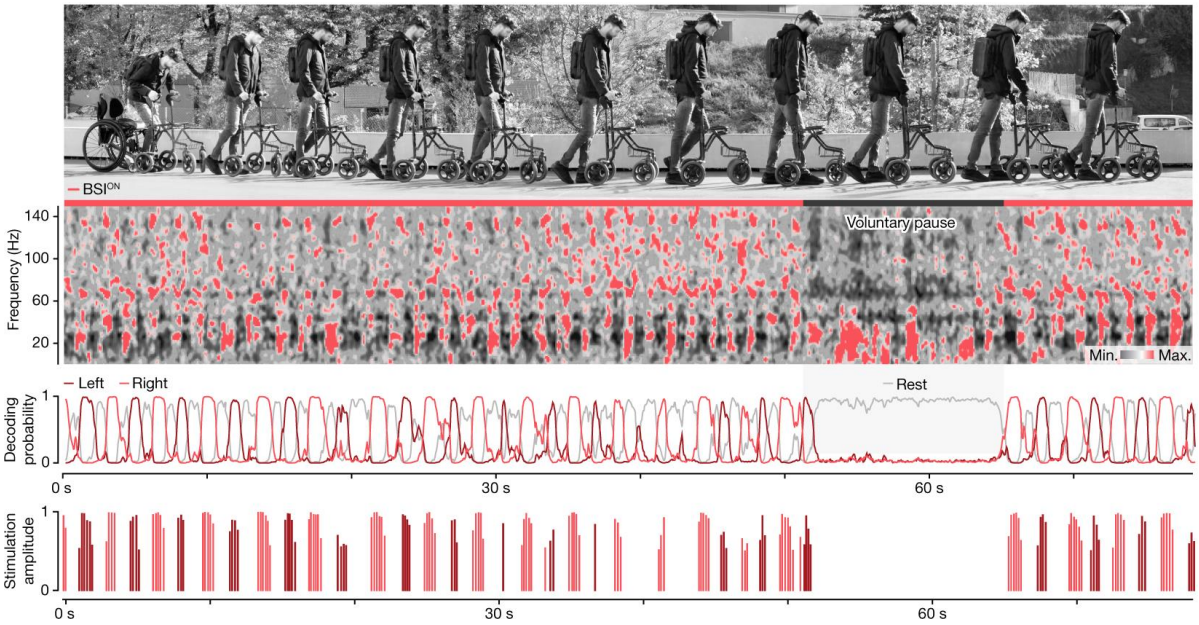


Figure 2.4: Performance of the Brain-Spine Interface during Gait Decoding, Adapted from [88].

A key strength of the approach was the robustness of the idle state, preventing false positives during voluntary pauses. Analysis of the feature distribution revealed that active states were characterized by high-frequency components, while idle states were dominated by low-frequency activity. Turning off the stimulator confirmed the causal role of the BSI. Overall, this work illustrates the feasibility of a high-accuracy, closed-loop system for restoring motion after spinal cord injury. A weakness of this prototype system was that the signal-processing was done on a portable PC running Matlab software. The work in this thesis was undertaken specifically to address this challenge and enable a portable, lower-power solution.

2.2 Hardware Implementations of BCIs: Architectures and Design Trade-offs

When considering embedded BCIs, the system must be miniaturized to be integrated into portable or implantable devices. For the latter, power must be limited to meet thermal

dissipation constraints, and more generally, reducing power extends battery life and enables long-term operation. Many BCI platforms have been proposed, including existing Microcontroller Units (MCUs) [174] and Digital Signal Processors (DSPs) [175]. General-purpose circuits have a power consumption that is too high for power-constrained miniaturized medical applications, as shown in an embedded implementation (Appendix B.3). There is thus a growing need for custom hardware solutions that can decode brain signals using minimal power, while also meeting other well-known metrics such as accuracy and low-latency.

2.2.1 Review of the State-of-the-Art of BCI Integrated Circuits

A typical block diagram of a BCI decoder is shown in Figure 2.5, which illustrates the main steps that the system implements between the recorded brain signals and the outputs sent to the effector. The decoder takes in N channels, sampled at a given Sampling Rate (SR) with a resolution of n bits and it outputs a classification as one of M classes with a given Decision Rate (DR). An intermediate feature extraction step is often applied before generating the outputs. In such cases, the decoder model can be split into two sub-modules: the feature extractor and the feature decoder. We assume that the features F are extracted at the same rate as the outputs (DR).

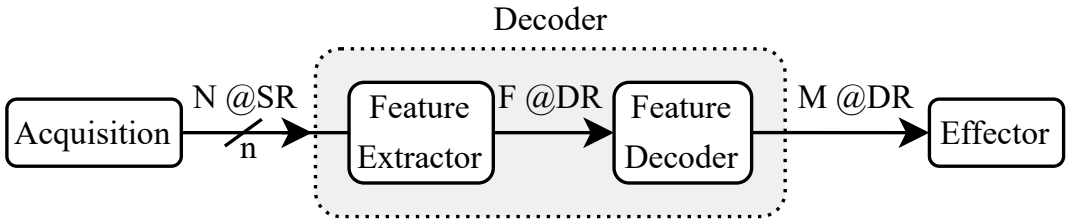


Figure 2.5: Reference Model for a BCI Decoder Showing a Two-Step Decoding with Feature Extraction.

We performed a review of the literature on hardware systems for brain signal decoding and identified papers presenting hardware systems that could be used for BCI motor decoding. The search was performed using PubMed, Scopus, Web of Science, IEEE Xplore and Google Scholar, and covered published work between 2010 and 2025, a period that witnessed a significant progress in chip development for BCI applications.

The reviewed works are listed in Table 2.2 where we show the type of signal that is decoded, the number of channels N, the input SR, the bit resolution n of the Analog-to-Digital Converter (ADC), the type of features, the extraction method, the number of features F, the type of task, the type of decoder, the number of output classes M, the DR, and the accuracy. When input (channel or feature) selection is applied, we report the number of selected inputs followed by the total number.

The system in [149] has two use cases, one where EEG signals are decoded for seizure and Parkinson disease tremor detection and one where ECoG signals are decoded for finger movement classification. We treat these two use cases separately, denoted as [149]a and [149]b, respectively. Similarly, two configurations are described in [111], which we refer to as [111]a and [111]b, with different numbers of channels (1536 and 3072, respectively). The circuit in [11] will also appear twice in the next section's graphs as in was tested for two decoding tasks (1-D and 2-D movement). Although the decoding system in [193] is tested for different applications, we only focus on its use in motor imagery.

Table 2.2: Summary of State-of-the-Art Brain Signal Decoding Hardware Implementations

Article		Inputs				Features			Task				
#	Ref.	Signal	Ch.	SR(Hz)	Bits	Type	Method	Nb.	Type	Decoder	Classes	DR(Hz)	Accuracy
1	2024 [193]	EEG	8/16	128	8	-	-	-	Mix[0]	CNN	2	0.5	Task related[(0)]
2	2024 [182]	EEG	8	250	10	-	-	-	Drone	TRCA	12	2.5	85.17%
3	2019 [90]	EEG	10/59	100	16	EB	Filter	3	Motor	CNN	3	1	80.5%
4	2017 [30]	EEG	1/14	128	21	TFC	CWT	1	Motor	Bayesian	2	1	80.5%/68.0%[(1)]
5	2016 [175]	EEG	4	250	10	EB	FFT	1/16	Eye state	Bayesian	2	0.5	100%
6	2011 [156]	EEG	4	256	12	EB	FFT	1	Seizure	Threshold	2	2	-
7	2022 [149]	Mix[(2)]	64/256	2000	10	Mix[(3)]	Custom	64	Mix[(4)]	DT	2/6	0.5-4	Task related[(4)]
8	2020 [93]	ECoG	32	260	4	EB	Analog	32	-	-	-	-	-
9	2019 [174]	ECoG	7/32	500	16	EB	Filter	1/14	Motor	Bayesian	2	1.33	87%
10	2016 [5]	ECoG	62	1000	12	PC	PCA	3	Motor	MLP	6	2	82.4%
11	2014 [180]	ECoG	7/32	1200	8	EB	DCT	42	Motor	Linear	2	3.33	82.9%
12	2010 [32]	ECoG	16	256	9	Mix[(5)]	Custom	-	Seizure	KNN	2	10	98.2%/97.8%[(6)]
13	2024 [144]	MEA	192/512	20000	10	DNC	Filter	128	Text	LDA	31	1	91.3%
14	2022 [11]	MEA	93	10000	16	SBP	Custom	93	Motor	SSKF	2/4	20	100%
15	2021 [187]	MEA	1024	20000	10	-	-	-	Motor	Threshold	7[(7)]	4.33[(7)]	85%[(7)]
16	2019 [111]	MEA	3072[(8)]	19300	10	-	-	-	-	-	-	-	-
17	2016 [23]	MEA	15	25000	16	-	-	-	Motor	SNN	4	4	70%
18	2015 [34]	MEA	40/128	10000	6	-	-	-	Motor	ANN	12	50	99.3%
19	2012 [125]	MEA	32	31250	8	-	-	-	Motor	Threshold	32	11.1	94%

[(0)] The circuit is tested for different applications including SSVEP, affect monitoring, and mental and motor imagery.
[(1)] Sensitivity (true positive) vs Selectivity (true negative).
[(2)] Tested on EEG epilepsy datasets, a μECoG array in vivo, and an ECoG finger movement dataset.
[(3)] The circuit can extract energy band features, phase values and Hjorth parameters.
[(4)] Epileptic seizure (95.6% sensitivity-96.8% selectivity), Parkinson tremor (82.6% sensitivity-78.4% selectivity) and finger movement (73.3% accuracy).
[(5)] Spatial cross-channel correlations, Time-domain and Frequency-domain features, Nonlinear chaotic values.
[(6)] Sensitivity (true positive) vs Specificity (true negative).
[(7)] No number of decoded classes is reported, neither a decision rate. We assume 7 classes (five for a 2D movement and two for button clicks), which gives a decision rate of 4.3 Hz to explain the reported information transfer rate (480 bits/min [115]) if we assume an accuracy of 85%.
[(8)] Two configurations with 1536 (resp. 3072) electrodes sampled at 19300 (resp. 18600) Hz.

MEA systems sample at a higher rate in order to detect spikes and they generally do not extract intermediate features, but instead, typically decode temporal characteristics (e.g. firing rate) of the spike train [144, 161]. However, the MEA decoder in [11] extracts Spiking Band Power (SBP) features, the average of a signal filtered in the 300-1000 Hz band. For ECoG and EEG, the most common extracted features are Energy Bands (EBs) that measure the signal energy at specific frequency ranges. Common frequency bands for these signals include δ(0-4 Hz), θ(4-8 Hz), α(8-13 Hz), β(13-30 Hz), and γ(30-100 Hz) bands [160]. ECoG signals can also have a high-γ (75-200 Hz) modulation during movement and speech. One approach to extract time-frequency components is to use a wavelet transform [50]. Another system [5] directly performs a PCA on the raw signals, and uses the obtained components as features for decoding. The number and type of features plays a key role in determining the decoding performance and energy consumption of the overall system.

Many of the existing hardware decoders (mainly for EEG and ECoG signals) presented in Table 2.2 are limited to two output classes and the decision rate (classifications per

second) is often below 5 Hz. Emerging motor control applications require more classes to control a higher number of Degrees of Freedom (DoFs), for multi-dimensional movements.

Most of the systems use linear decoders and Bayesian models. The system in [182] performs a TRCA, which is an end-to-end decoding method where the signals are matched with templates tailored to the output classes. The circuit in [11] uses a Steady-State Kalman Filter (SSKF) to decode a 1-D or 2-D motor task. Decision trees are another approach for classification, which was used by [149]. Some of the systems use neural network approaches: [193], [90], [23] and [34] directly decode the raw input signals, whereas, [5] uses a MLP to decode PCA-based extracted from ECoG signals. All systems report accuracies that are typically over 80%, although the decoding difficulty varies with the different tasks.

It is important to note that all the systems, in Table 2.2, that perform decoding, use a model that has been trained offline. Two circuits [23, 182] are, however, able to update the model coefficients based on error feedback. Overall, the State-of-the-Art on BCI circuits shows that the approaches for brain signal decoding remain highly heterogeneous, suggesting there is not yet a consensus on the best approach.

2.2.2 Optimizations for BCI Circuit Design

Despite the heterogeneity of BCI systems, some techniques have been commonly used by designers to optimize power consumption, when implementing on-chip decoders. In this section, we present key power optimizations that have been used in the state-of-the-art brain decoding circuits. These can be grouped into three main categories: (i) input selection and reduction, (ii) compute, and (iii) circuit-level optimizations. We focus on optimizations that reduce the number of operations or their power consumption.

2.2.2.1 Input Selection and Reduction

Electrode Selection

A highly effective technique to reduce power consumption is to only collect data from electrodes that provide useful information for the current decoding task. The circuit in [149], uses a 256-electrode grid to decode stimulation patterns. In the training mode, four modules are used to convert the signals to a digital format so they can all be used by an offline training algorithm. This algorithm determines the probabilistic weights of a DT decoder, and a subset with the 64 best electrodes (selected using a 16×16 switch matrix) to be used when performing each step of the online classification. With this approach, the circuit benefits from the potential information from a large number of electrodes, while limiting the power required for signal acquisition and decoding, as three of the four modules are powered-off. Similarly, the MiBMI circuit [144] and the Neuralink chip [187] both include a switch matrix to dynamically select input channels for decoding. The diagram in Figure 2.6 shows an acquisition system for which a selection algorithm dynamically selects a subset of electrodes for inference.

Electrode selection has also been studied for online training of BCI algorithms. A method based on penalization is introduced in [105] to obtain sparse decoding models when using a Recursive Exponentially Weighted N-way Partial Least Squares algorithm for training. The algorithm is based on a PARAFAC tensor decomposition [53] where the objective function to minimize is the quadratic error between the labels and predictions. With this penalized method, an additional cost term, proportional to the number of

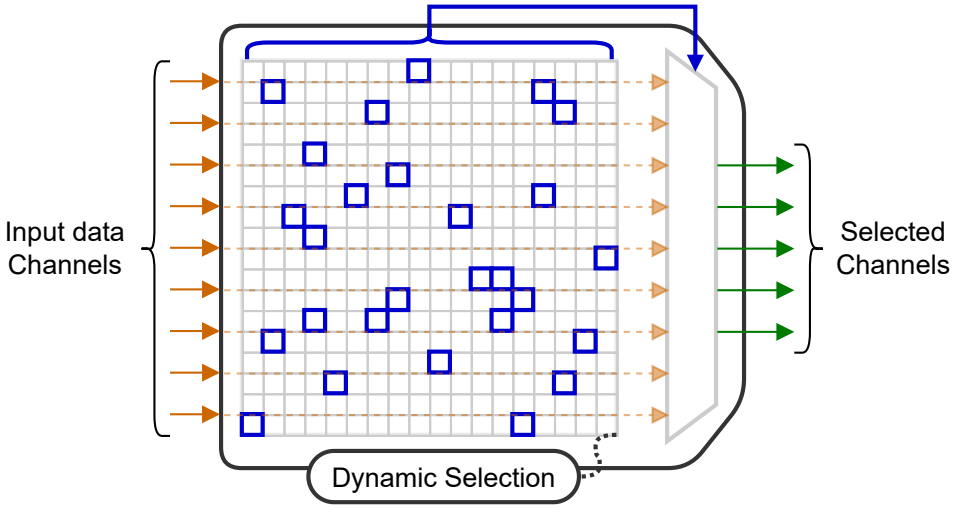


Figure 2.6: Block Diagram of a Switch Matrix for Channel Selection.

electrodes, is added to the objective function. By applying this penalization, a model with 75% sparsity was obtained, meaning that only one out of every four electrodes was required. The sparse model achieved a cosine similarity close to that of the original model, thus reducing compute time and memory consumption with no significant loss in accuracy.

Other techniques can be used for electrode selection. The authors in [174] select a subset of the available electrodes that maximizes the contrast between the classified states. By applying this technique, they were able to empirically select the seven best channels out of the 32 available to be used for decoding. Other criteria can be used for channel selection. For instance, the Fisher criterion was used in [180] to select seven channels out of 32 to be used for decoding. The authors in [193] and [30] select the best channels as the ones achieving the highest decoding accuracy for a specific task. Finally, the selection can also be based on previous works that have determined spatial localization of task-related brain signals, such as was done in [90]. More precisely, in [193], only eight electrodes are selected to be placed at specific positions on a head-strap to support a variety of mental tasks. Then, a subset of these electrodes is used for each task, which minimizes the amount of input data to process, hence the power consumption.

Although less common, it is possible to combine inputs from different electrodes such as the approach in [23] where data from 15 different channels is merged to create a combined spike train. This optimization makes it possible to use a reduced number of synapses per neuron in the downstream SNN decoder.

Feature Selection and Reduction

Input selection is not limited to the raw input signals, but can also be applied for the extracted features. When we refer to *feature selection*, it means that certain features that were initially identified, are completely eliminated. When *feature selection* is applied, there is a double benefit: the eliminated features no longer need to be computed, which reduces energy, and the size of the downstream decoder is reduced. *Feature reduction* consists of combining the computed features into a reduced intermediate representation that is used by the decoder. However, the energy savings are limited to the decoder level as all the features are computed and then combined. Both techniques are common in the BCI literature, and they can be combined with electrode selection techniques. Figure 2.7 shows a diagram of a general circuit that uses a *feature selection* module to extract features

adapted to a given task. The dynamically chosen features are then combined (projected in a latent space) using a *feature reduction* block.

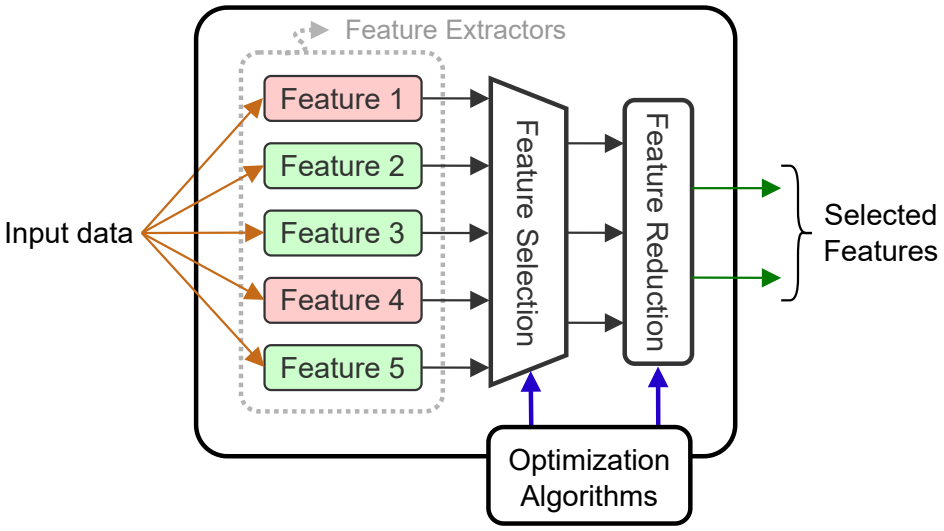


Figure 2.7: Block Diagram of a Feature Extraction Block Supporting *Selection* and *Reduction*.

The authors in [174] perform *feature reduction* using a class-wise PCA (retaining 92% of variance per class) to extract linear combinations of two selected EBs (one is in the α and β bands, the other in the high-γ band). These linear combinations are then reduced to a single value using LDA, and this value is fed to a Bayesian decoder for ECoG motor (grasp) decoding. In earlier work, the same authors [175] also implemented a similar approach on a DSP, however, the PCA was done across all the data, using a higher threshold (99.7%) for the retained variance.

The epileptic seizure detection System-on-Chip (SoC) in [32] has a hardware module to extract multiple types of features, followed by a dimension reduction unit that combines the extracted features for inference. The reduction unit is programmable and could implement a PCA that has been computed offline. The *feature reduction* step contributes to the low-power implementation, and enables the classification using a micro-processor.

In [30], a different approach is used based on an adaptive Wavelet Transform (WT). The key idea is to generate, during the training phase, a special template that identifies the onset of movement intent. For each of the two classes, the Fast Fourier Transform (FFT) of the input signal is calculated. Using the Fisher Discriminant Ratio, they identify a small set of frequency bands that best discriminate between the classes. The Inverse Fast Fourier Transform (iFFT) of these selected frequency bands is computed, thus providing a template which is used for detection. Using this approach for designing a custom filter, requires less power than extracting each of the required frequency components, which is an interesting approach for *feature selection*. The Fisher criterion is also used in [180] to select the six best frequency bands for a decoder using a Discrete Cosine Transform (DCT) to extract EB features.

Another adaptive method is applied in [93] where the circuit extracts power envelopes in the gamma band using band-pass filters. Since the specific frequency range varies between individuals, a dual-mode architecture is proposed with two operating regimes: Full-Band (FB) and Base-Band (BB) modes. In the FB mode, the circuit captures the raw brain signals in the analog front-end layer with high resolution (8-10 bits sampled at

13 kHz), and the digital back-end uses these signals to compute patient specific weights, for filters in the neural pre-processing unit. For inference, the system operates in the BB mode with lower ADC resolution (3-4 bits sampled at 260 Hz). In this mode, power-band features are extracted using the previously calibrated filters.

In addition to reducing the number of features, off-chip processing can help in selecting the set of extracted features when the circuit offers a configurable feature extraction engine. As the circuit in [149] is meant to be used for different applications, a set of varied features is extracted from the selected electrodes. These consist of: Line Length (LL), Local Motor Potential (LMP), High-frequency Oscillations (HFO), Hjorth parameters (Activity, Mobility and Complexity), PAC, Phase Locking Value (PLV) and EB extracted using band-pass filters. The authors reduce power using an energy-aware objective function for training that minimizes cross-entropy while also penalizing the use of features requiring a high power consumption. This method reduces power by 64% compared to a standard cross-entropy baseline, while resulting in less than a 2% loss in decoding accuracy.

The circuit in [144] extracts Distinctive Neural Code (DNC) features to decode handwritten characters. These features correspond to spike rates at different time windows measured from different electrodes. An offline algorithm selects the 64 DNCs that best distinguish a given class, and these are fed into a LDA decoder for classification. *Feature selection* reduces the number of operations by a factor of 320× and the memory size by 7.8×. The optimizations can be combined with other methods to further reduce the power consumption of BCI systems.

Conclusion 2.2.2.1

Electrode selection focuses on choosing the most informative channels, while *feature selection* and *reduction* retains the most relevant representations. When *feature selection* is applied, there is a double benefit: the eliminated features no longer need to be computed, and the size of the downstream decoder is reduced.

2.2.2.2 Compute Optimization

As seen above, reducing the number of electrodes and the number of features directly reduces the number of compute operations, thus overall power. However, the power also depends on the way these operations are implemented in hardware. We review some techniques that have been used to efficiently implement specific compute operations.

Approximate Compute Operations

Another approach to reduce power consumption is to reduce the precision of the decoding computations, often using approximate computing techniques. For example, to implement the feature extraction methods efficiently in hardware, the authors in [149] approximate the theoretical definitions for the features, with simplified equations. For example, the L_2-norm is replaced by L_1 when computing the Hjorth Parameters (HPs), and the phase is computed using a linear arctangent approximation with a Lookup Table (LUT) error correction. These simplifications yield features similar to the theoretical ones computed in Matlab (with a median correlation above 0.9) but at a lower compute cost. Similarly, the authors in [125] use a low-power spike pattern matching based on a logical *AND* comparison between the spike counts instead of a costly multiplication.

The circuit in [5] implements a simplified PCA algorithm, described in [33], which uses

only adders and multipliers. The norm and division operators are simplified to additions and multiplications using fixed-point arithmetic. In addition, a log-sigmoid activation function is implemented using a LUT.

Moving from floating-point to fixed-point arithmetic is common in low-power decoding circuits [11, 90, 149, 156]. The circuit in [180] achieves a 82.9% (resp. 75.7%) accuracy when using fixed-point (resp. floating point) operations. The authors suggest that, in some cases, the quantization for fixed-point arithmetic can remove random ECoG noise, thus yielding a better decoding accuracy. In addition to using fixed-point arithmetic, the chip in [187] supports custom precision to save power. They report a "99.7% match" compared to the original approach with floating-point arithmetic.

More system-level approximations can be applied to further reduce the power consumption. For example, the circuit in [193] implements a teacher-student CNN approach for decoding. It consists of two CNNs with different architectures: the teacher, a large model with high power consumption and high decoding accuracy, and the student, a smaller model that consumes 70% less energy, but achieves 14% lower accuracy. During the decoding process, only the student model is turned on when the same state is decoded. The system switches between both models only when a state transition is detected with a low confidence level. This level is defined according to a confusion matrix between the decoders' outputs. The hybrid architecture reduces the overall energy consumption by 55%. The system also takes advantage of the EEG signal sparsity for further power reduction.

Merging Operations

Another technique to reduce the number of compute operations is to merge multiple operations in the decoding equations. In other words, certain steps can be reordered or combined before being implemented in hardware.

An interesting combination method is described in [182] where SSVEP are decoded using a TRCA algorithm. The processing combines three main steps: pre-processing using a temporal filter, feature extraction with spatial filtering, and pattern recognition with a SSVEP template signal. The steps are mapped into a single matrix for fast computation. In this paper, the decoder's updates are directly applied to the combined matrix, based on a feedback signal.

In [180] a DCT is applied to extract the signal energy at given frequencies. The DCT of a signal $\{x_n,\ n \in [\![0, N-1]\!]\}$ is defined by Equation 2.1,

$$X_m = \sum_{n=0}^{N-1} x_n \cos\left[\frac{m\pi}{N}\left(n+\frac{1}{2}\right)\right] \tag{2.1}$$

where m is a scaling parameter fixing a frequency and N the length of the input signal. To reduce the number of multiplications, the authors propose a reduced-resolution quantization of the cosine function in equation (2.1). In fact, by using 11 levels of quantization ($\leq$4 bits), the system only requires 11 multiplications to compute the DCT, regardless of the size of the input signal. More precisely, the input signal samples are divided (using a LUT based on their indexes) into 11 sets and the elements of each set are summed before multiplying with the corresponding quantized cosine coefficient. This technique can be more generally applied to any large sum of products, assuming one of the factors can be coarsely quantized, and it was later extended, in this thesis, to the Continuous Wavelet Transform (CWT) in Section 3.2.

Conclusion 2.2.2.2

Compute-level optimizations aim to reduce the compute complexity of arithmetic operations using either exact or approximate simplification methods.

2.2.2.3 Circuit-level Optimizations

In addition to input and compute optimizations, some circuits propose circuit-level improvements. These can include analog implementations of the compute operations or other custom low-power circuit techniques.

Analog Approaches

In BCI systems, input signals are analog. While digital circuits offer greater flexibility and ease of design than their analog counterparts, they require multi-channel, low noise ADCs at a sampling rate ranging from a few hundreds to a few thousands of Hz to provide the digitized input data. Analog circuits can reduce power either by extracting features before digitization or by enabling efficient decoding accelerators.

A typical processing task is energy extraction over one or several frequency bands. Performing this function in the analog domain reduces the amount of information that is sent to the decoding stage, and hence reduces the power consumption. For instance, a tunable analog band-pass Gm-C filter has been used to capture gamma-band envelopes consuming only 1.05 μW/ch [93] including 0.205 μW/ch for feature extraction. Another design integrates per-band energy estimation with feedback control, reporting 56 μW/ch [86]. In [189], the authors use analog modules to extract the signal energy over four frequency bands, consuming 3.1 μW/ch, and 10.8 μW/ch when accounting for bias, clocks and amplifiers. In [84], an analog integrate-and-fire circuit was used to encode signal energy into a blinking LED rate, achieving 0.74 μW/ch [84] for SBP detection.

Analog approaches can also be used in accelerators for brain signal decoding to achieve higher computing power efficiency. In [34], the authors propose an analog decoder, consisting of a dense neural network implemented using current-controlled oscillators. Operating in the sub-threshold region, the oscillator matrix consumes 0.4 μW. Looking forward, neuromorphic architectures aim to replicate the behavior of biological synapses and neurons on silicon, achieving power-efficient sparse computations. In [182], an analog decoding accelerator is implemented using a memristor array, consuming 2.46 mW, to perform the matrix-vector multiplications of the TRCA algorithm applied to a 12-class decoding task. The system can be incrementally updated by integrating feedback responses in a closed-loop system to maintain performance over time. Moreover, the authors in [23] propose an end-to-end neuromorphic decoder that decodes four classes using spikes and consumes about 4 mW in their experiments. The last two circuits serve as proof of concept for neuromorphic architectures in future adaptive low-power BCIs.

Other Power Optimizations

Circuit optimizations can also involve custom blocks with specific properties. For example, a custom Static Random Access Memory (SRAM) in [156] using a sub-threshold 6T design is used with the decoder. It provides an ultra-low-power retention mode during which it has a leakage power of only 28 fW/bit. This circuit also focuses on improving the efficiency of the power delivery system (DC/DC converter). The MiBMI circuit's [144] power consumption is reduced by an additional factor of 12.9× through memory sharing by using variable size memory blocks across inputs and classes. Clearly, such methods can

be combined with the compute and algorithmic techniques from the previous sections.

Conclusion 2.2.2.3

Circuit-level optimizations provide fine control over energy and performance by customizing hardware at the lowest abstraction level. Comparatively fewer researchers have explored such techniques as they require access to a custom silicon design flow.

2.2.2.4 Optimization Summary

We have analyzed all of the studied circuits to identify which of the different power optimization techniques were employed and the result is summarized in Table 2.3, where for each of the circuits used for brain signal decoding, we list the applied optimizations, the power consumption, and the implementation. The summary shows that the most common optimization techniques used in BCI circuits are those based on input selection, feature selection and reduction, and compute approximation. The input selection and reduction methods often consist of using an offline algorithm to only select inputs that are relevant. Compute approximations, such as quantization of the numeric values and the use of simplified functions, are also widely used. Fewer works have employed circuit optimization techniques as these require specialized design skills.

Table 2.3: Summary of BCI Circuit State-of-the-Art Optimizations

	Article		Selection/Reduction		Compute Optim.		Circuit-level		Power	Implem.
#	Ref.	Signal	Channel	Feature	Approx.	Merge	Analog	Custom	(mW)	
1	2024 [193]	EEG	✓		✓				0.0905	ASIC
2	2024 [182]	EEG				✓	✓		2.46	ASIC
3	2019 [90]	EEG	✓		✓				25	FPGA
4	2017 [30]	EEG	✓	✓					47.3	FPGA
5	2016 [175]	EEG		✓					34	DSP
6	2011 [156]	EEG			✓			✓	0.00099	ASIC
7	2022 [149]	Mix	✓	✓	✓				0.453	ASIC
8	2020 [93]	ECoG		✓			✓		0.0346	ASIC
9	2019 [174]	ECoG	✓	✓					150	MCU
10	2016 [5]	ECoG			✓				152	FPGA
11	2014 [180]	ECoG	✓	✓	✓	✓			0.72	FPGA
12	2010 [32]	ECoG		✓					0.23	ASIC
13	2024 [144]	MEA	✓	✓				✓	0.88	ASIC
14	2022 [11]	MEA			✓				0.581	ASIC
15	2021 [187]	MEA	✓		✓				24.7	ASIC
17	2016 [23]	MEA	✓				✓		4[(1)]	ASIC
18	2015 [34]	MEA					✓		N/A[(2)]	ASIC
19	2012 [125]	MEA			✓				0.537	FPGA

[(1)] The reported power only takes into account the decoding step.
[(2)] The authors report a power consumption of 0.4μW for the implemented layer, they do not include the acquisition nor the output combination cost.

2.3 Metrics for System Comparison and Limitations of Current Systems

In the literature on algorithms for brain signal decoding, the primary metric of interest is usually the classification or decoding accuracy. For real-time motor applications, it is also important that the systems achieve a sufficiently high decision rate to meet the latency requirements of the target application. In order to compare different decoding systems, we consider the simplified reference "decoder" model (Figure 2.5). We define the Input Data Rate (IDR) as the total input bits per second in Equation 2.2. Note, for circuits that perform channel selection, N is the number of selected channels.

$$IDR[bits/s] = N \times n[bits] \times SR[Hz] \tag{2.2}$$

2.3.1 Analysis based on Output Classes per Second

Controlling a system with a large number of DoFs requires a decoder with more output classes, and of course, achieving high accuracy with a large number of output classes is more difficult. To capture both the task difficulty and the latency requirement, we propose a metric called Classes per Second (CpS), defined in Equation 2.3.

$$CpS[classes/s] = M[classes] \times DR[Hz] \tag{2.3}$$

In Figure 2.8, we present a scatter plot of the systems (log-log scale), where the horizontal axis corresponds to the IDR and the vertical axis corresponds to the output CpS. The shapes correspond to the tested application, the numbers in the shapes refer to the paper references in Table 2.2, and the colors correspond to the type of signal being decoded. Despite the systems being highly heterogeneous, we note an increasing tendency for the decoded CpS as function of the IDR.

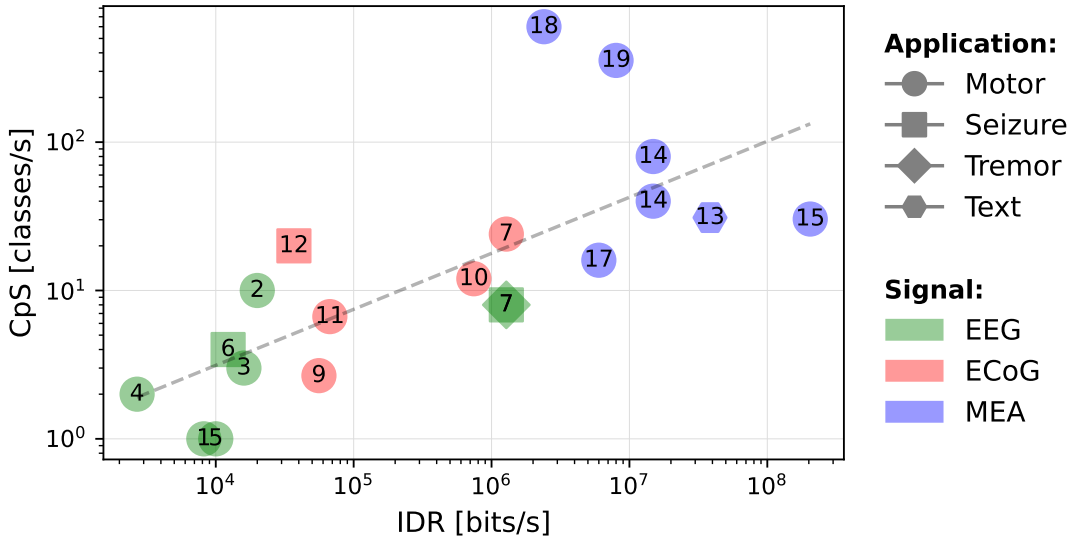


Figure 2.8: Output Classes per Second as a Function of the Input Data Rate (bits/s). The Numbers Represent the References in Table 2.2.

A least-squares fit trendline has been plotted. The regression yields a strong positive correlation (R-value = 0.849 for a Spearman's test with a p-value of 0.000002). The

circuits presented in [34] (#18) and [125] (#19) are outliers, achieving a high CpS value as they are able to decode a high number of classes (12 and 32) at a high DR (11.1 and 50 Hz) using highly parallel compute operations. The equation for the trendline is given in Equation 2.4.

$$CpS = 0.097 \times IDR^{0.38} \tag{2.4}$$

For a given application, the required DR and the number of classes are known, thus the CpS can be easily calculated. The required IDR can be estimated using our trendline, and can be used to ensure the number of electrodes, the SR and ADC accuracy of the acquisition system are sufficient. Of course, there are limitations for each parameter that must be respected: for example there is no benefit in increasing the SR beyond the intrinsic frequency of the input signals.

Conclusion 2.3.1

The trendline shown in Figure 2.8 provides a reference for designers of BCI motor systems to estimate the required IDR for decoding a given number of CpS.

2.3.2 Information Transfer Rate to Measure Decoding Performance

The ITR, described in [121], is a metric frequently used in BCI systems to describe the amount of information being extracted, taking into account the number of classes M, the interval between decisions $1/DR$ (the maximum time to produce one new classification for real-time applications), and the achieved accuracy P. The formula for computing ITR is shown in Equation 2.5.

$$ITR[bits/min] = 60 \times DR \times \left(\log_2 M + P \log_2 P + (1-P) \log_2 \left(\frac{1-P}{M-1} \right) \right) \tag{2.5}$$

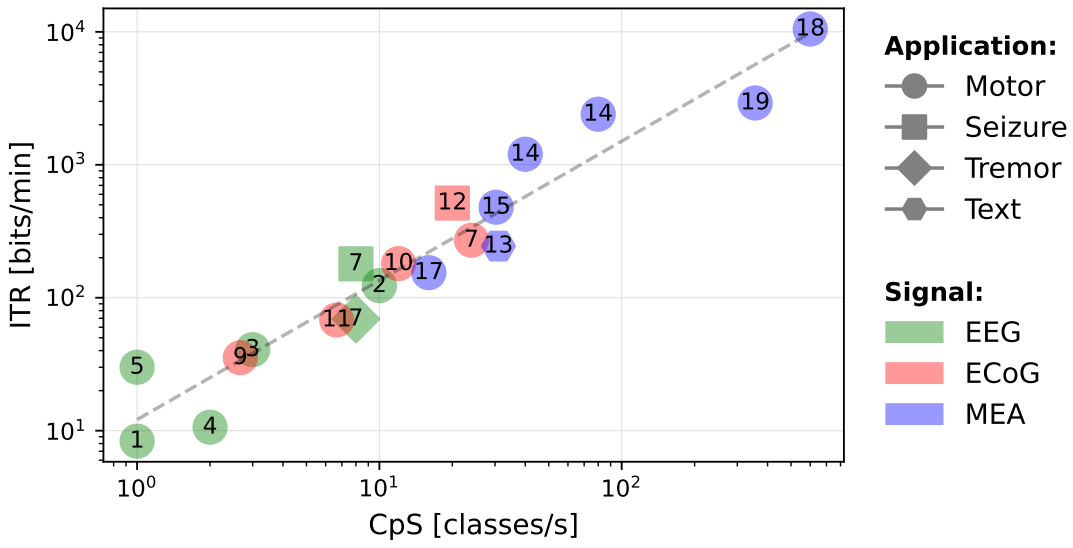


Figure 2.9: Scatter Plot of the Information Transfer Rate (bits/min) as a Function of the Classes per Second.

Our hypothesis is that systems are not significantly differentiated by accuracy, but that accuracy is mainly a requirement for the system usability. This suggests that, once the accuracy requirement is met for a given task, a high ITR reflects a high DR and number of classes M, thus a high CpS. In Figure 2.9, we present a scatter plot of ITR versus CpS. It clearly shows that the two metrics are highly correlated (R-value = 0.97 for a Spearman's test with a p-value of 10^{-11}), showing that accuracy has a marginal influence on the ITR of BCI systems. We will consider the ITR as a metric that reflects the CpS value to compare performance, while validating a sufficient level of accuracy that makes a system usable.

Conclusion 2.3.2

This trend shows that the ITR is mainly dominated by the DR and the number of classes, which define the CpS metric, and that an acceptable level of accuracy ($\geq$70%) is rather a requirement than a differentiating metric when comparing BCI decoders.

2.3.3 Analysis of Circuit Power Consumption

The CpS and the ITR metrics depend on the rate at which information is output, but they do not provide insight into the computational complexity required to perform the decoding. Given the diversity of decoding approaches, both in terms of the algorithms and the numeric formats, it is difficult to analytically determine the computational cost of an algorithm, for example by counting the number of arithmetic operations. The power consumption of a decoding circuit thus provides a proxy measurement of the computational cost of the decoding technique.

2.3.3.1 Power vs IDR

A scatter plot of the power consumption (in mW) of the circuits versus the IDR is shown in Figure 2.10. The plot shows that non-motor applications (epileptic seizure and Parkinson Disease (PD) tremor) using EEG and ECoG signals (grey box) consume less power than those decoding motor intentions. This may be due to the smaller number of features, which require fewer compute operations. For example, only the β-band is extracted in [156] (#6) and only the high-γ band in [93] (#8). For motor decoding, the circuits can be separated into three groups according to the type of signal. For EEG (green box) and ECoG (red box), the power consumption does not show a strong correlation with the IDR. This implies that the power consumption is determined by the type of processing, including feature extraction, rather than the amount of input data.

It is interesting to note that the reported power consumption of the EEG and ECoG decoding circuits is quite similar, while the ECoG circuits decode a much higher IDR. This might be explained by the fact that EEG signals are averaged over a larger number of neurons than ECoG signals [150], thus requiring more processing to extract relevant information.

In MEA systems, the IDR is significantly higher due to a higher SR. The plot also shows that the power consumption scales with the IDR for these systems. As MEA systems are often based on spike detection and sorting [125, 164, 187, 191], the main variable is the number of inputs or the sampling rate, which determines the IDR. This power dependency could also be due to the MEA signals requiring less computation than

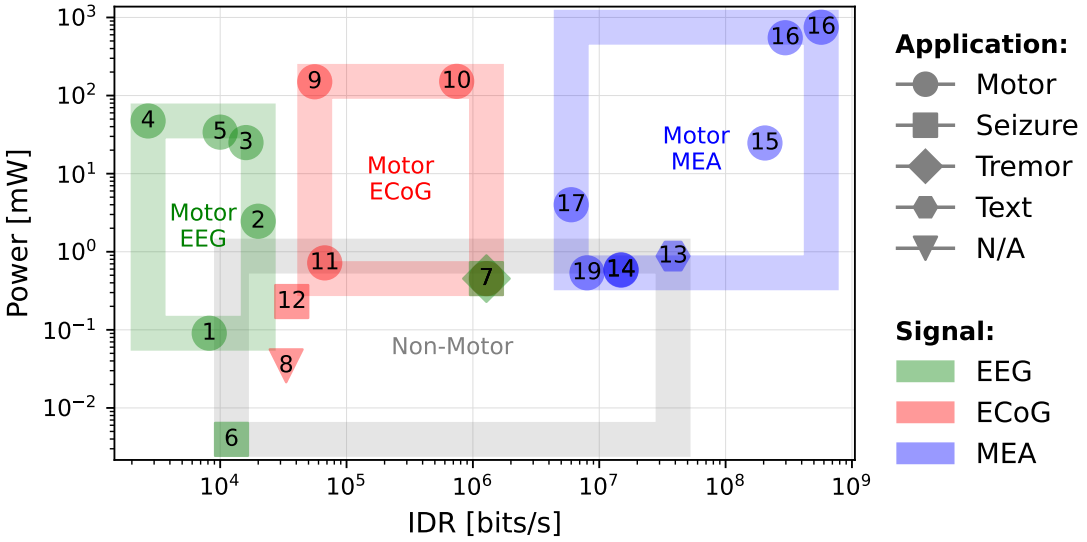


Figure 2.10: Power Consumption (mW) as a Function of the Input Data Rate (bits/s). Non-Motor Decoding Circuits Are Grouped Together (grey), and Motor Decoding Circuits Are Grouped According to the Signal Type: Green for EEG, Red for ECoG, and Blue for MEA.

ECoG and EEG, which means that the majority of the power is used for data acquisition or transmission, parameters that directly scale with the IDR. As the field evolves, and new circuits appear, it will be seen if this trend continues.

2.3.3.2 Power per Channel vs ITR

The ITR characterizes the amount of useful information output by the system, considering the DR, number of output classes M, and the decoding accuracy P. As discussed in Section 2.3.2, the ITR primarily reflects the classification rate, as the accuracy of the studied circuits does not vary significantly. The computational cost of an algorithm can be indirectly measured by the overall power consumption. However, the power also depends on the number of channels, especially for systems using MEAs. To better understand the scaling of the power consumption, we consider a normalized metric, the Power per Channel (PpC), to take into account this effect. For embedded applications, the aim is to minimize the circuit power consumption, hence the PpC, given a fixed number of electrodes required to extract the necessary spatial information.

Figure 2.11 shows a scatter plot of the PpC versus the ITR. The best performing circuits are towards the bottom right side of the plot, meaning they provide the most information while using the least power per electrode. In this plot, we note that the circuits in the bottom right region use MEAs. There is no clear difference in the performance between ECoG and EEG systems and the PpC is comparable for both types of signals. In fact, for such systems, the performance depends more on the decoding approach than the signal type. For instance, the circuits that exclusively extract EB features [30, 90, 174, 175] (#3,4,5,9) have a lower performance than those extracting a wider range of features [32, 149] (#7,12), or directly decoding raw signals [182, 193] (#1,2).

We plot the trendline obtained using a least-squares fit on these circuit metrics. A Spearman's test shows a negative correlation (R-value = -0.602 with a p-value of 0.008) between the ITR and the PpC metric. A way to look at this, is by considering a system

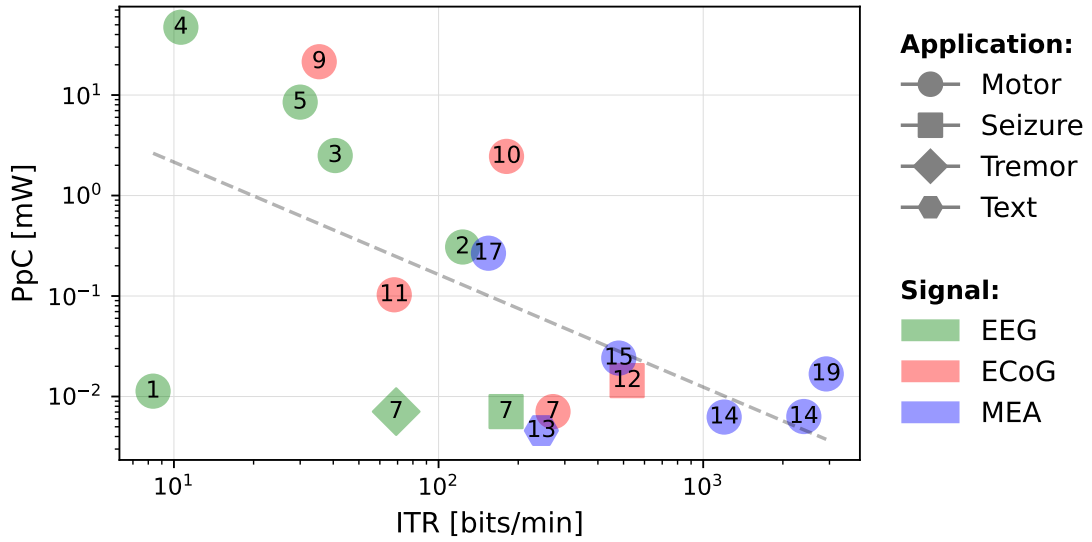


Figure 2.11: Power per Channel (mW) as a Function of the Information Transfer Rate (bits/min).

that doubles the number of electrodes to increase its ITR. If the power consumption also doubles, the system will only move horizontally on this plot. Whereas, we see that in actual systems, when the ITR doubles, the PpC tends to decrease. One explanation for this trend is that there is always some fixed system energy cost that does not scale with the number of electrodes and hence, as the number of channels increases, this fixed cost is amortized, resulting in a lower PpC. This suggests that systems with a large number of channels can simultaneously benefit from hardware sharing and a large amount of input data, which can enable decoding more classes or potentially improving accuracy. Furthermore, for a fixed number of input electrodes, the best performance is achieved by circuits that extract the most useful information (ITR) with minimal computational cost (PpC). The system [193] (#1) is an outlier as the authors' main objective is to reduce the power consumption without necessarily improving the ITR. This plot provides a high-level method to compare BCI decoding circuits, where the best performance is in the lower, right region.

Conclusion 2.3.3.2

The negative correlation between ITR and PpC suggests that it is possible to improve the ITR while simultaneously reducing the PpC, showing that both algorithmic performance and energy consumption can be optimized together in BCI systems.

2.3.4 Limitations and Problem Statement: Bridging the Gap in BCI using Hardware-Software Co-Design

Despite the progress in BCI system development, from improving decoding algorithms to reducing power, there remains a gap between algorithmic requirements and hardware capabilities of such systems. At the algorithmic level, advanced feature extraction techniques, such as wavelet transforms and spatio-temporal correlations provide information that can significantly improve decoding accuracy. However, implementing these opera-

tions in hardware requires high compute capacity, making their deployment in embedded or implantable systems constrained by the power budget.

Existing approaches optimize power consumption through input selection and reduction, approximate arithmetic, or custom hardware design. As discussed in 2.3.3, the energy consumption of current systems is driven by the processing complexity rather than the IDR, highlighting the importance of feature extraction pipelines in determining the efficiency of a system. Based on these findings from our literature review, we chose to focus the thesis work on reducing the compute and power requirements of feature extraction while preserving high-quality representations for decoding.

Problem Statement

Designing power-constrained BCI systems requires a cross-layer approach that simultaneously addresses the compute needs and the power budget. The objective is to co-design algorithms and hardware architectures to extract relevant information while meeting the requirements for embedded and implantable BCIs.

Chapter 3

Low-Power Feature Extraction using the Approximate Wavelet Transform

Chapter 2 highlighted that the efficiency of a Brain-Computer Interface (BCI) system is primarily determined by the approach used to extract meaningful information. Among existing techniques, the Continuous Wavelet Transform (CWT) is a commonly used method to extract time-frequency features for brain signal decoding. However, it is compute intensive, which can limit its use in portable or implantable applications. Addressing this limitation requires reformulating the problem in a way that takes into account energy vs. accuracy trade-offs.

This chapter introduces two approximate methods for computing the CWT, based on piecewise approximations of the wavelet function. We show that these methods significantly reduce the number of required operation, thus the power consumption, without compromising the usability of the extracted features in BCI decoding tasks. In addition, we derive an energy model to estimate the expected savings at the algorithmic level, allowing an early validation of design choices before proposing a hardware implementation.

- Section 3.1 reviews existing exact and approximate Wavelet Transform (WT) methods, and discusses their limitations, especially when used in BCI systems.
- Section 3.2 introduces two proposed approximate CWT methods, including their mathematical formulation and algorithmic implementation.
- Section 3.3 analyzes the compute complexity of the exact and proposed techniques. Energy models are also derived to estimate the energy consumption of the different methods.
- Section 3.4 describes the datasets and the metrics used to benchmark the proposed approximate CWT approaches. The feature extraction algorithms are evaluated in a BCI decoding pipeline.
- Section 3.5 shows the achieved energy savings and the impact on decoding accuracy as the result of the approximations. It discusses the trade-off results and their implications for low-power BCI design.

Parts of this chapter have been presented in the 13th International Winter Conference on Brain-Computer Interface, and published in [135].

3.1 Existing Methods for Computing the Wavelet Transform

The WT, more generally known as the CWT, is used to extract time-frequency components from input signals, through a convolution with a time-localized frequency pattern. This method has been shown to extract useful information in the context of BCIs. A variant, the Discrete Wavelet Transform (DWT), offers lower compute cost, but it is subject to constraints that limit its applicability in certain BCI systems. Prior attempts to reduce the compute cost of the CWT, have approximated the wavelet globally with polynomials or with piecewise splines. These works focused on reducing computational cost but did not evaluate how the proposed approximations affect decoding performance in BCI applications. The remainder of this section describes these methods and discusses their limitations.

3.1.1 Continuous Wavelet Transform Fundamentals

A wavelet ψ is a zero-mean, time-localized oscillating pattern. Figure 3.1 illustrates three commonly used wavelets: Morlet, Mexican Hat, and Haar wavelets.

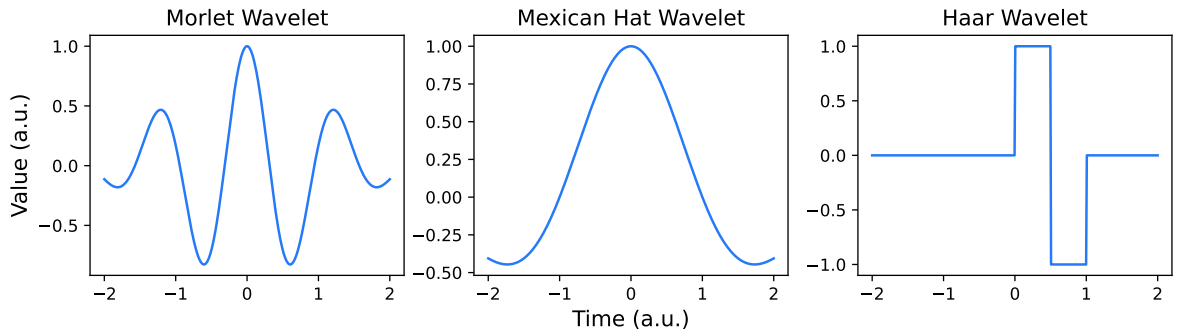


Figure 3.1: Examples of Three Common Wavelets: Morlet, Mexican Hat, and Haar.

These patterns are called mother wavelets, and serve as prototypes to derive a family of wavelets by scaling. The resulting wavelets correspond to oscillations at different frequencies, localized in the time domain. Formally, the scaled version of ψ is defined as:

$$\psi_a(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t}{a}\right) \tag{3.1}$$

where $a > 0$ is the scale parameter used to derive the scaled wavelet. As shown in (3.1) the wavelet is often normalized by a factor of $1/\sqrt{a}$ to preserve its overall energy across different scales. Smaller scales correspond to compressed wavelets, which capture high-frequency oscillations with fine temporal precision, while larger scales correspond to dilated wavelets, detecting low-frequency components over a larger time window. The ability to extract information in both the time and frequency domains makes wavelets well suited for analyzing non-stationary signals such as neural recordings.

The CWT uses the scaled wavelets ψ_a and a translation parameter $\tau \in \mathbb{R}$ to decompose a signal into frequency components that are localized in time [50]. Given a real input signal x and a mother wavelet ψ, the wavelet transform component at scale $a > 0$ and time shift

τ is defined as the inner product of x with the scaled and translated complex-conjugated wavelet [55], as follows:

$$S(a,\tau) \;=\; \int_{-\infty}^{+\infty} x(t)\,\psi_a^*(t-\tau)\,dt \;=\; \int_{-\infty}^{+\infty} x(t+\tau)\,\psi_a^*(t)\,dt \tag{3.2}$$

Note, although this definition is that of an inner product, it can also be expressed as a convolution with a time-reversed wavelet. For convenience, we will refer to this operation as a convolution throughout the manuscript. Assuming the wavelet has finite support K, the integral in Equation (3.2) is reduced to a finite interval, as shown in Equation (3.3).

$$S(a,\tau) \;=\; \int_{K} x(t+\tau)\,\psi_a^*(t)\,dt \tag{3.3}$$

3.1.1.1 Discrete-Time Formulation

In practice, input signals are sampled with a finite resolution, and processing such signals requires the continuous formulation of the CWT to be adapted to discrete time. Let X denote the sampled signal, and Ψ_a be the discretized version of the scaled wavelet ψ_a. The transform in (3.2) can be written as a finite sum:

$$S(a,\tau) \;=\; \sum_{i=0}^{N-1} X[i+\tau]\,\Psi_a^*[i] \tag{3.4}$$

where N is the length of the discretized support of the wavelet, and $\tau \in \mathbb{Z}$ the discrete time shift. The CWT can be seen as sliding dot products between the signal and the scaled wavelets, and it can be used to get the energy in a given frequency band. Such spectral band power characteristics, extracted at different scales a, can be used for BCI applications.

3.1.1.2 Direct Convolution Method

A direct way to compute the CWT is to evaluate the discrete-time expression in Equation (3.4). For each scale a, the scaled wavelet Ψ_a is slid across the signal and each dot product at a shift τ corresponds to a single point of the convolution output.

For a signal of length S and a wavelet of length N, the number of operations per convolution is proportional to $\mathcal{O}(S \cdot N)$. When using complex wavelets, the convolution involves both real and imaginary components, leading to $2\times$ more operations than for real wavelets. Exploiting wavelet symmetry or anti-symmetry can reduce the number of multiplications by half through factorization.

Conclusion 3.1.1.2

The direct convolution method remains computationally expensive, as its cost grows linearly with both the signal length and the wavelet support.

3.1.1.3 FFT-Based Convolution Method

A more efficient approach consists in computing the convolution as a multiplication in the frequency domain. Using the convolution theorem, the CWT can be obtained by applying a Fast Fourier Transform (FFT) to the input signal, multiplying the spectral

result by the pre-computed FFT of the wavelet, and then applying an Inverse Fast Fourier Transform (iFFT) to obtain the convolution in the time domain. The result of the FFT-based method is identical to that of the direct convolution, up to numerical rounding errors.

For a signal of length S and a wavelet of length N, the required FFT length is $M = S + N$, leading to an algorithmic complexity of $\mathcal{O}(M \log M)$. Typically, for a given input signal, the energy at different scales is evaluated. Since the forward FFT of the signal can be reused for all scales, the per-convolution cost is reduced, as the cost per additional scale is that of an additional multiplication in the frequency domain and an iFFT. This makes the FFT-based approach well suited for large-scale feature extraction pipelines. The method has been used to efficiently compute CWT, due to its its low complexity [13, 127]. Many hardware implementations of the FFT algorithm exist [47, 54], some of which have used it to evaluate the CWT [16].

Conclusion 3.1.1.3

The FFT-based method achieves the same accuracy as direct convolution, while reducing the algorithmic complexity. It is even more efficient when extracting multiple scales, as the compute cost of the forward FFT can be amortized across multiple frequency bands.

3.1.2 Discrete Wavelet Transform (DWT)

The CWT enables the extraction of time-frequency components at arbitrary frequencies and time positions. However, this flexibility comes at the cost of high computational requirements. To reduce this cost, the DWT [55, 147] was introduced as a discretized version of the CWT. In the DWT, both scale and time shift are restricted to dyadic values: $a = 2^j$ and $\tau = k \cdot 2^j$, with $j, k \in \mathbb{Z}$. This restriction results in a logarithmic partitioning of the time-frequency plane. The obtained family of dyadic wavelets is defined as:

$$\psi_{j,k}(t) \;=\; 2^{-j/2}\,\psi(2^{-j}t - k) \tag{3.5}$$

In the CWT, both the scale a and the translation τ are continuous, which can result in a redundant representation: close values for scales correspond to very similar frequency components, and nearby time shifts correspond to overlapping time windows. The DWT reduces this redundancy by discretizing these parameters, while producing enough coefficients to represent the signal without duplication. With orthogonal wavelets such as Daubechies [171], the family forms an orthonormal basis, which guarantees a perfect signal reconstruction from the DWT coefficients.

3.1.2.1 Efficient Implementation using Cascaded Filters

The dyadic structure of the DWT allows an efficient implementation using cascaded filter banks. At each stage, the signal is split in two parts using a low-pass filter (approximations) and a high-pass filter (details), followed by down-sampling by two. The procedure is applied recursively on the approximation branch to derive the DWT components. The cost is reduced from $\mathcal{O}(S \cdot N)$ for a direct CWT to $\mathcal{O}(S)$ for a signal of length S. Hardware implementations of such filter-bank structures have been proposed for one-dimensional [64] and two-dimensional [184] DWTs. In particular, 1D designs are used for time signal processing, while 2D implementations are commonly applied in image

processing, for wavelet-based compression. These implementations confirm that the DWT is suitable for hardware implementation.

3.1.2.2 Limitations of DWT in BCI Applications

Despite its efficiency and its ability to extract sufficient information for certain applications [28], the DWT has important limitations in the context of BCI. The first limitation comes from the dyadic partitioning of the scaling parameter. Frequency bands are constrained to power-of-two multiples of a base frequency. In practice, brain signals are modulated in specific frequency bands that are not necessarily log-spaced. The restricts the use of DWT when particular non dyadic frequencies are needed.

A second limitation is that the DWT requires orthogonal wavelets to be efficiently implemented. This constraint excludes non-orthogonal wavelets often used in brain signal processing, such as the Mexican Hat [25, 79] and Morlet wavelets [14, 20, 87, 88, 140]. These wavelets, used in BCIs, cannot thus be combined with the DWT method and therefore require implementing the CWT technique.

Conclusion 3.1.2

Although the DWT is efficient and well suited for hardware, its restrictions limit its use in BCIs, when extracting specific non dyadic frequency components or when using non-orthogonal wavelets.

3.1.3 Existing Approximate Methods

Several approaches have been proposed to reduce the computational cost of the CWT by approximating the wavelet function. For that, approximation techniques aim to replace the exact form of the wavelet with a simplified representation for which the convolution requires fewer operations. These approximations can lower the number of operations compared to the exact CWT, but their impact on feature extraction and decoding performance in BCI systems has not been rigourously studied.

3.1.3.1 Polynomial Approximations

One approach to approximate the CWT is to replace the wavelet with a polynomial function [118]. By using a polynomial of degree p, it is possible to reduce the algorithmic complexity of the CWTto $\mathcal{O}(p{\cdot}N)$, by limiting the number of multiplications. Significantly increasing the degree of the polynomial can be necessary to improve the fit (match the extrema and flatten the tails). While this technique reduces the arithmetic cost, the quality of the approximation decreases rapidly if the degree of the fitted polynomial is not sufficient.

As illustrated in Figure 3.2, low-degree polynomials fail to capture all the details of the wavelet, whereas higher degree polynomials match the shape better at the cost of higher compute complexity when evaluating the convolution. Two different approximations are shown: a regression-based fit and an interpolation through selected extrema. The regression provides a global least-squares optimal fit but misses local variations, while interpolation through extrema forces the polynomial to exactly pass by key points of the wavelet with the risk of larger oscillations between constraints.

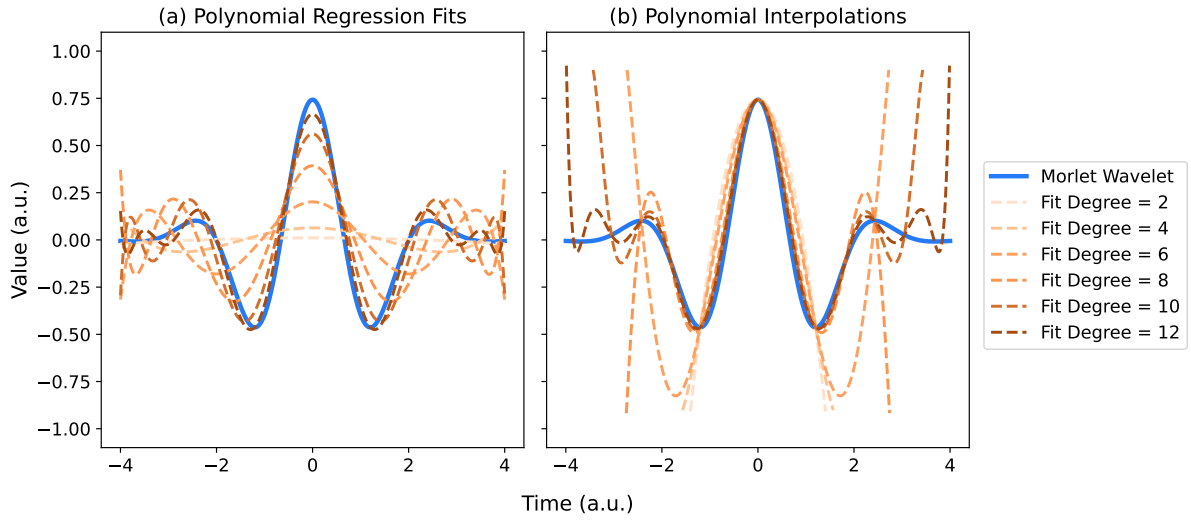


Figure 3.2: Polynomial Approximations of the Morlet Wavelet. (a) Regression Fits from `np.polyfit`. (b) Interpolation Fits Passing through Extrema.

3.1.3.2 Spline-Based Approximations

Wavelets can also be approximated by splines, which are defined as piecewise polynomial segments [89]. As splines are low-degree polynomials, using them can similarly limit the number of multiplications when computing the CWT. However, unlike polynomials defined on the entire interval, splines confine the approximation error locally, since each point influences only a limited part of the approximated function. In practice, cubic splines are commonly used, as they provide continuity up to the second derivative while keeping the computational cost low.

Figure 3.3 shows two approaches: a cubic spline interpolation [42], where the wavelet is constrained to pass through given knot values, and a cubic B-spline least-squares fitting [167], where the wavelet is approximated by a linear combination of cubic basis functions. Increasing the number of knots/control points reduces the approximation error.

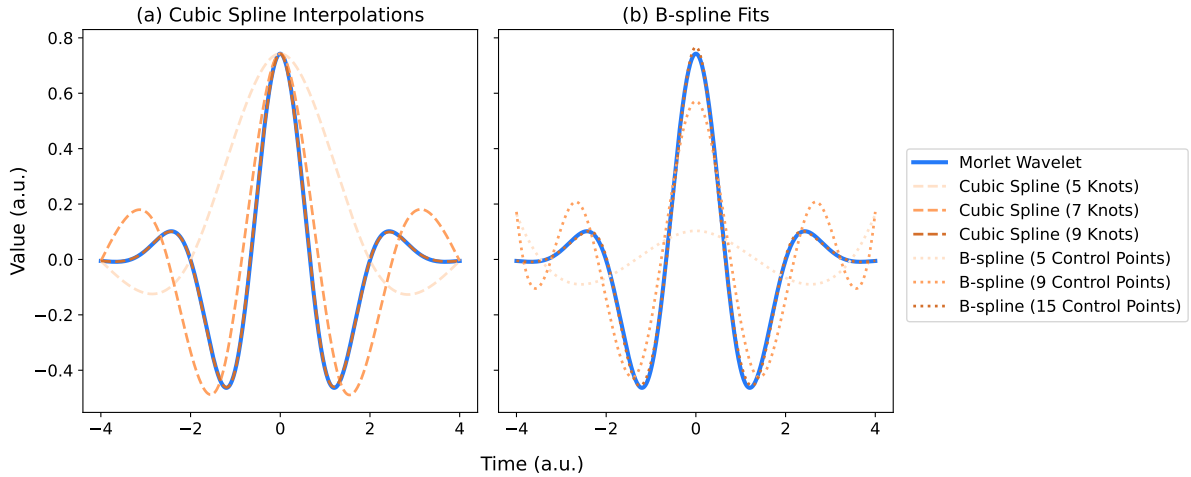


Figure 3.3: Spline Approximations of the Morlet Wavelet. (a) Cubic Spline Interpolations with Different Numbers of Knots. (b) B-Spline Least-Squares Fits with Different Numbers of Control Points.

Spline-based methods provide a better approximation than global polynomials. These approaches theoretically reduce the arithmetic cost of the CWT [12, 112] as they reduce the number of operations.

Conclusion 3.1.3

Although approximate methods can reduce the arithmetic cost of the CWT by limiting the number of multiplications, their impact on feature quality and decoding performance in BCI applications has not been evaluated.

3.2 Proposed Approximate Methods for CWT

In this section, we present two methods to compute the CWT using piecewise polynomial approximations of the wavelet. The wavelet ψ_a in Equation (3.2) is replaced by its piecewise approximation ϕ defined in Equation (3.6)

$$\forall i \in [1, J-1], \quad \phi(t) = \phi_i(t), \quad t \in [\alpha_i, \alpha_{i+1}], \tag{3.6}$$

where $(\alpha_i)_{i=1}^{J}$ is a J-point subdivision of the wavelet interval, and each ϕ_i is a polynomial of fixed order k defined in Equation (3.7).

$$\phi_i(t) \;=\; \sum_{j=0}^{k} p_{i,j}\, t^j \tag{3.7}$$

3.2.1 Approximation using k-Order Polynomials

The key idea of this technique is to transform the convolution into a sum of per-segment integrals that can be simplified by repeatedly applying integration by parts a number of times equal to the polynomial order k. Starting from Equation (3.3), by replacing ψ_a by its approximation ϕ and applying integration by parts once, we obtain:

$$S(\tau) \;=\; \Big[X^{(1)}(t+\tau)\,\phi(t)\Big]_{\alpha_1}^{\alpha_J} \;-\; \int_{\alpha_1}^{\alpha_J} X^{(1)}(t+\tau)\,\phi'(t)\,dt \tag{3.8}$$

Repeating this integration k times yields the general expression:

$$\begin{aligned} S(\tau) \;=\; & \sum_{j=0}^{k-1} (-1)^j \Big(X^{(j+1)}(\alpha_J+\tau)\,\phi^{(j)}(\alpha_J) - X^{(j+1)}(\alpha_1+\tau)\,\phi^{(j)}(\alpha_1) \Big) \\ & +\; (-1)^k \int_{\alpha_1}^{\alpha_J} X^{(k)}(t+\tau)\,\phi^{(k)}(t)\,dt \end{aligned} \tag{3.9}$$

where $X^{(j)}$ denotes the j-th antiderivative of the signal and $\phi^{(j)}$ the j-th derivative of the wavelet approximation. Since ϕ is a piecewise polynomial of order k, its k-th derivative $\phi^{(k)}$ is a constant on each segment, and can therefore be factored out of each segment of the integral in Equation (3.9).

3.2.1.1 Approximation using Third-Order Splines

Cubic splines are commonly used, as they preserve continuity up to the second derivative, as discussed in Section 3.1.3.2. To better understand how the method introduced in Equation (3.9) can be applied to cubic splines, we choose $k = 3$. This leads to the following expression:

$$\begin{aligned} S(\tau) = \; & X^{(1)}(\alpha_J + \tau)\,\phi_{J-1}(\alpha_J) \; - \; X^{(1)}(\alpha_1 + \tau)\,\phi_1(\alpha_1) \\ & - \; X^{(2)}(\alpha_J + \tau)\,\phi'_{J-1}(\alpha_J) \; + \; X^{(2)}(\alpha_1 + \tau)\,\phi'_1(\alpha_1) \\ & + \; X^{(3)}(\alpha_J + \tau)\,\phi''_{J-1}(\alpha_J) \; - \; X^{(3)}(\alpha_1 + \tau)\,\phi''_1(\alpha_1) \\ & + \; \sum_{i=1}^{J-1} \phi'''_i \int_{\alpha_i}^{\alpha_{i+1}} X^{(3)}(t + \tau)\,dt \end{aligned} \tag{3.10}$$

where $X^{(j)}$ is the j-th antiderivative of the signal x, and ϕ'''_i is the (constant) third derivative of the spline segment on $[\alpha_i, \alpha_{i+1}]$.

The example illustrates that, for a cubic spline approximation, the input must be integrated successively up to four times to evaluate the integral of $X^{(3)}$. Such repeated integrations of digital signals may be problematic, as numerical integration accumulates quantization errors, which can result in a highly inaccurate final approximation [38, 113, 117]. Furthermore, storing all of the higher-order integrals requires additional storage.

Conclusion 3.2.1.1

High-order splines offer good approximations of the wavelet, but they remains impractical for low-cost computation of the CWT.

3.2.1.2 Piecewise Linear Approximation

Given the challenges of high-order spline approximations, a simpler, yet practical, approach is to approximate the wavelet with linear segments (k=1). By aligning the subdivision points with the extrema of the wavelet, the piecewise linear approximation preserves its main oscillations with only a few coefficients. Figure 3.4 illustrates the principle of this piecewise linear approximation.

After integrating by parts, the compact formulation of the CWT calculated using the piecewise linear approximation is shown below:

$$\begin{aligned} S(\tau) \; &= \; X^{(1)}(a_J + \tau)\,\phi_{J-1}(a_J) \; - \; X^{(1)}(a_1 + \tau)\,\phi_1(a_1) \; - \; \sum_{i=1}^{J-1} \phi'_i \int_{a_i}^{a_{i+1}} X^{(1)}(t + \tau)\,dt \\ &= \; X^{(1)}(a_J + \tau)\,\phi_{J-1}(a_J) \; - \; X^{(1)}(a_1 + \tau)\,\phi_1(a_1) \; - \; \sum_{i=1}^{J-1} \phi'_i \Big[X^{(2)}(t + \tau) \Big]_{\alpha_i}^{\alpha_{i+1}} \\ &= \; X^{(1)}(a_J + \tau)\,\phi_{J-1}(a_J) \; - \; X^{(1)}(a_1 + \tau)\,\phi_1(a_1) \; - \; \sum_{i=1}^{J} b_i\, X^{(2)}(a_i + \tau) \end{aligned} \tag{3.11}$$

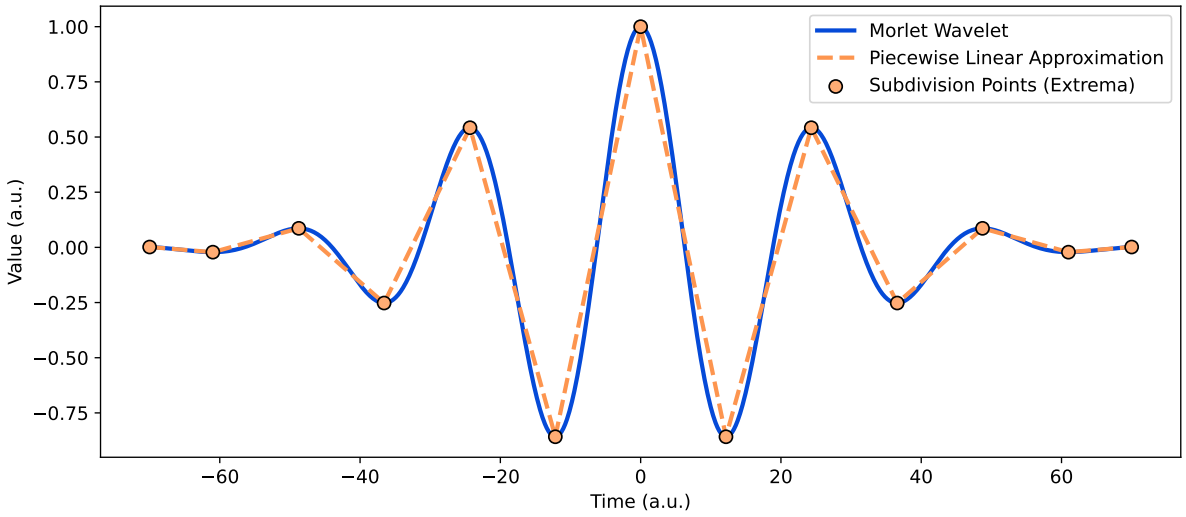


Figure 3.4: Piecewise Linear Approximation of the Morlet Wavelet.

with the re-ordered coefficients given as follows:

$$\begin{cases} b_1 = -\phi'_1 \\ b_i = \phi'_{i-1} - \phi'_i \quad \text{for } i \in \{2, \ldots, J-1\} \\ b_J = \phi'_{J-1} \end{cases} \tag{3.12}$$

The resulting technique, requires only the first and second antiderivatives of the signal, which reduces complexity compared to cubic splines. In addition, by limiting the calculation to the second antiderivative, we minimize the accumulation of quanitzation errors. In the following, we denote this linear piecewise formulation as the Piecewise Linear Continuous Wavelet Transform (PLCWT).

Conclusion 3.2.1

The PLCWT approximates the wavelet by linear segments, and limits the number of multiplications to J, by grouping signal samples with the same quantized wavelet value. Unlike spline-based approximations, it requires only the first and second antiderivatives of the signal. Reducing the number of antiderivatives limits the accumulation of quantization error and the required memory storage.

3.2.1.3 Piecewise Constant Approximation (Quantization)

An even simpler approach is to directly approximate the wavelet by constant values over each subdivision interval, which corresponds to the case of k=0. A similar idea has previously been applied to compute a Discrete Cosine Transform (DCT) for movement decoding [180], but here we extend the concept to the CWT. This piecewise constant approximation quantizes the wavelet, preserving only a small number of values. Figure 3.5 shows an example of the Morlet wavelet with $Q = 5$.

If the wavelet is quantized into Q discrete levels $(v_q)_{q=1}^{Q}$, let $\mathcal{I}_q$ denote the set of indices of the wavelet samples that are approximated by v_q. The CWT in Equation (3.4) can then be evaluated by factorizing the signal samples that are multiplied by the same wavelet level, which reduces the number of multiplications to Q. This results in Equation (3.13).

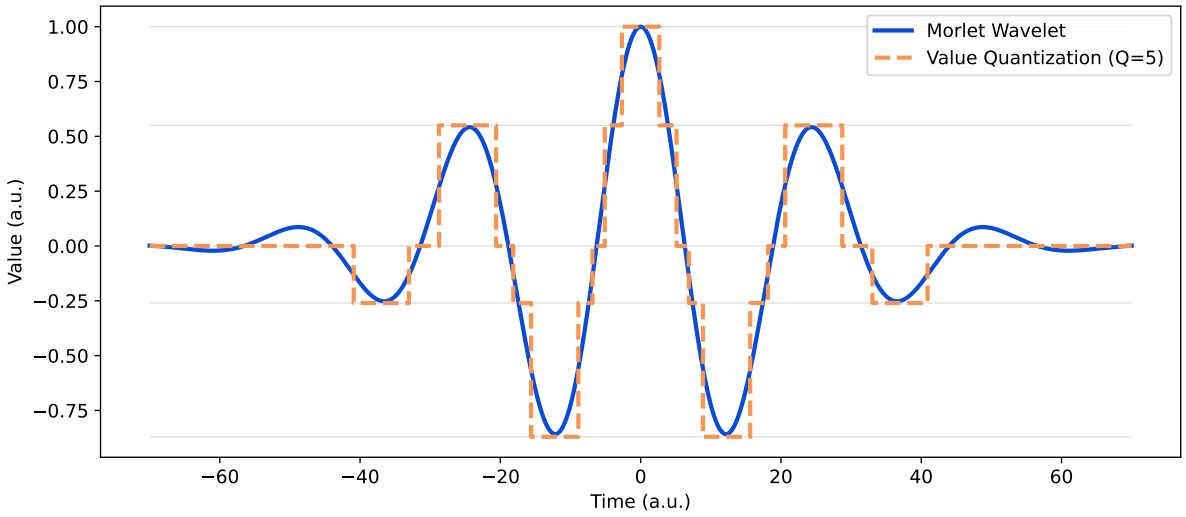


Figure 3.5: Piecewise Constant (Quantized) Approximation of the Morlet Wavelet

In the following, we denote this method as the Quantized Continuous Wavelet Transform (QCWT).

$$S(\tau) \;=\; \sum_{q=1}^{Q} v_q \times \sum_{i \in \mathcal{I}_q} X[i+\tau], \tag{3.13}$$

Note that each non-zero quantization value may appear as either $+v_q$ or $-v_q$, since signal samples grouped under the same quantization level can contribute through addition or subtraction. In practice, only unique absolute values are counted among the Q quantization levels.

Conclusion 3.2.1.3

The QCWT approximates the wavelet by constant segments. It limits the number of multiplications to Q, by grouping signal samples with the same quantized wavelet value.

3.2.2 Linear Output Correction and L_2-Norm Energy

Approximating the wavelet introduces a systematic error in the CWT output, since the wavelet details are not perfectly preserved. This error can be partially compensated by a linear correction, as previously proposed in [27]. The idea is to rescale and offset the approximate convolution as shown in Equation (3.14):

$$S_{\text{correct}}(\tau) \;=\; \alpha\, S(\tau) + \beta, \tag{3.14}$$

where α and β are parameters obtained offline through a least-squares linear regression between exact and approximate CWT outputs, for each frequency band. The correction is inexpensive to apply, as the multiplicative factor α can be absorbed directly by the approximation coefficients: the constant values v_q for the QCWT, and the coefficients b_i for the PLCWT. The additive term β requires only S additions per signal of length S, which is negligible compared to the overall convolution cost.

Energy-based features (obtained through the L_2-norm) are widely used in BCI systems [14, 20, 25, 62]. When such features are extracted, the correction must be applied to both the real and imaginary parts of the convolution before computing the magnitude. This step limits the error of the computed features, to maintain decoding performance.

Conclusion 3.2.2

Linear correction partially compensates the error introduced in the approximate CWT through a one-time calibration, with a low compute cost.

3.3 Complexity Analysis and Energy Models

The computational cost of each CWT method depends on the required number of arithmetic operations. Figure 3.6 summarizes the methods analyzed in this section.

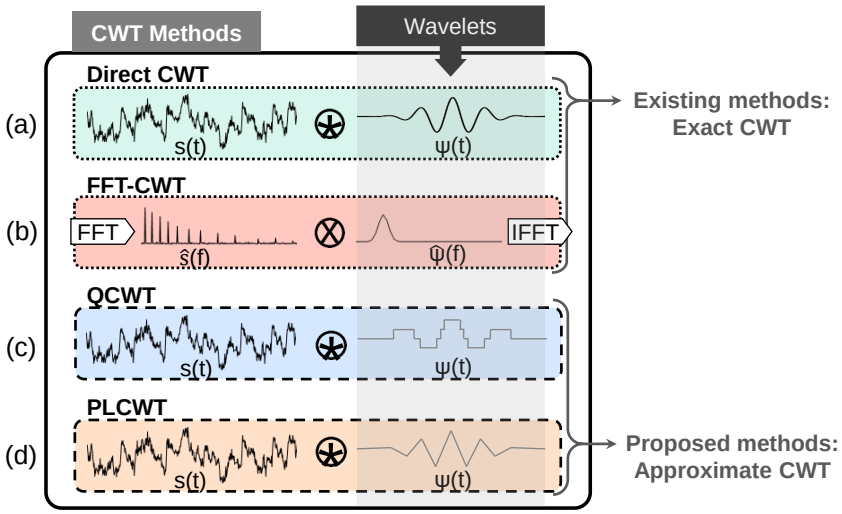


Figure 3.6: Overview of the Four CWT Methods: (a) Direct CWT in the Time Domain, (b) FFT-Based CWT in the Frequency Domain, (c) QCWT with a Piecewise Constant Approximation, and (d) PLCWT with a Piecewise Linear Approximation.

The existing exact approaches include the Direct CWT in the time domain and the FFT-based CWT, while the proposed QCWT and PLCWT methods approximate the wavelet to reduce compute operations. To compare the existing and proposed methods, we first provide an exhaustive count of additions and multiplications for the four approaches. We then discuss how the arithmetic precision impacts this cost, based on an energy model from [60]. Finally, we illustrate the scaling of these costs in BCI scenarios, with different buffer sizes, approximation quality, and number of frequency bands.

3.3.1 Operation Count

The complexity of each CWT implementation can be estimated by counting the number of elementary arithmetic operations. We distinguish between additions and multiplications, as they have different hardware costs. A complex multiplication requires four real multiplications and two real additions, while a complex addition requires two real additions. For a real-by-complex multiplication, only two real multiplications are needed. We denote by S the signal length, N the wavelet length, F the number of extracted frequencies, Q

the number of quantization levels for the QCWT, and J the number of subdivision points for the PLCWT. In the following, we provide the detailed operation counts for each of the four methods. We assume a complex wavelet convolved with a real input signal.

3.3.1.1 Direct CWT

The direct implementation of the CWT evaluates the discrete convolution in Equation (3.4) for each scale and time shift. The convolution at a single position τ requires N real-by-complex multiplications and $(N-1)$ complex additions. Since this is repeated for all S positions and F frequency bands, the number of operations is multiplied by $S \times F$. If the wavelet is symmetric or antisymmetric, half of the multiplications can be eliminated by factorizing signal points before multiplying by the corresponding wavelet coefficient.

3.3.1.2 FFT-CWT

The FFT-based method consists of three main steps: the forward FFT of the signal, the multiplication by the pre-computed FFT of the wavelet in the frequency domain, and the iFFT for each frequency band. For a signal of length S and a wavelet of length N, the required FFT length is $M = S + N$. In some cases, where only the convolution points corresponding to an overlap are needed, M can be restricted to S. An example is provided in Appendix B.1.2. Each FFT/iFFT operation requires $(M/2)\log(M)$ multiplications and $M\log(M)$ additions [68]. $F \times M$ complex multiplications are needed in the frequency domain. Note that the cost of the forward FFT is amortized across all frequency bands, making the method particularly efficient when F is large. When the wavelet is sparse, the number of frequency-domain multiplications can be reduced proportionally to the sparsity s as described in Appendix B.1.2. By defining the density $\rho = 1 - s$, the number of complex multiplications in the frequency domain is thus $\rho \times F \times M$.

3.3.1.3 QCWT

In the QCWT, the wavelet coefficients are quantized into Q discrete levels, as described in Section 3.2.1.3. As a result, the number of real-by-complex multiplications is reduced from N to Q compared to the Direct CWT, while the number of complex additions remains $(N-1)$ for a convolution at a shift τ. For F frequency bands and S convolution points, the number of operations scales with $S \times F$.

3.3.1.4 PLCWT

In the PLCWT, the wavelet is approximated by J linear segments, as introduced in Section 3.2.1.2. As shown in Equation (3.11), the convolution at each shift τ requires $(J+2)$ real-by-complex multiplications and $(J+1)$ complex additions, in addition to computing the first and second antiderivatives $X^{(1)}$ and $X^{(2)}$ of the signal. The cost of computing these antiderivatives is amortized across all frequencies. For S positions and F frequency bands, the total number of operations scales with $S \times F$, as in the Direct CWT and QCWT. As with the Direct CWT, wavelet symmetry can further reduce the number of multiplications by a factor of two.

3.3.1.5 Summary

Table 3.1 summarizes the operation counts for the four methods, including multiplication reductions when using wavelet symmetry or sparsity. These expressions provide a basis for the energy models that will be used to compare the methods.

Table 3.1: Number of Operations for Existing and Proposed CWT Methods, with and without Exploiting Optimizations such as Wavelet Symmetry or Sparsity.

Method	Additions	Multiplications	Multiplications (Optimized)
Direct CWT	$2 \times S \times (N-1) \times F$	$2 \times S \times N \times F$	$S \times N \times F$
FFT-CWT	$3(F+1)M \log M + 2FM$	$2(F+1)M \log M + 4FM$	$2(F+1)M \log M + 4\rho FM$
QCWT	$2 \times S \times (N-1) \times F$	$2 \times S \times Q \times F$	$2 \times S \times Q \times F$
PLCWT	$2 \times S \times (J+1) \times F$	$2 \times S \times (J+2) \times F$	$S \times (J+2) \times F$

3.3.2 Energy Models

The expressions in Table 3.1 are used to compute the required energy by assigning a reference energy cost to each operation. We select the widely-used values reported in [60], for a 45 nm technology, which provide energy per operation for a given arithmetic precision. Table 3.2 reports the estimated energy per operation assuming 32-bit precision.

Table 3.2: Energy per Operation at 45 nm for 32-Bit Precision, from [60].

Operation	Fixed-point	Floating-point
Addition	0.1 pJ	0.9 pJ
Multiplication	3.1 pJ	3.7 pJ

Fixed-point operations are generally less costly than floating-point operations. For 32-bit precision, a multiplication is 31× (resp. 4×) more costly than an addition in fixed-point (resp. floating-point) arithmetic. The total energy is computed as follows.

$$E_{\text{total}} = N_{\text{add}} \times E_{\text{add}} + N_{\text{mult}} \times E_{\text{mult}}. \tag{3.15}$$

Only arithmetic operations are considered, memory and register accesses are excluded. This serves as a first high-level step to evaluate the cost of the different methods before hardware implementation.

3.3.3 Analysis for Different Scenarios

In the following, we analyze how parameters such as buffer size, wavelet frequency, approximation quality, and multi-band configurations influence the compute and thus the energy cost in a BCI system.

3.3.3.1 Impact of CWT Parameters on Operation Count

We first evaluate how the buffer size and wavelet frequency impact the computational cost of the four CWT methods, assuming a sampling frequency of 590 Hz [99]. For each method, we report the number of additions in Figure 3.7 and multiplications in Figure 3.8.

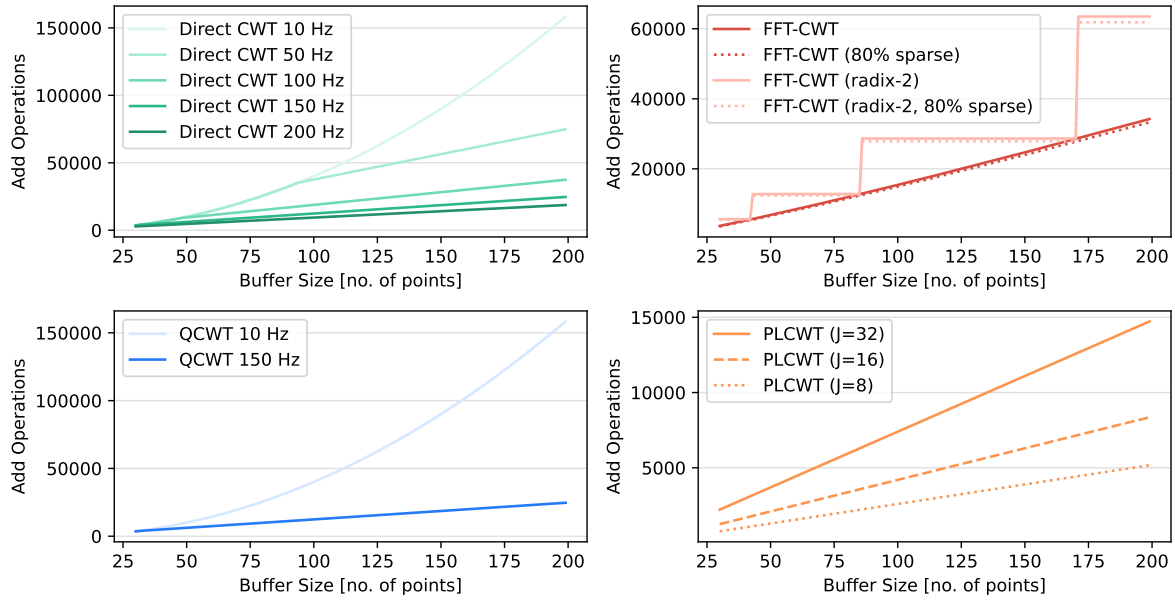


Figure 3.7: Number of Additions as a Function of Buffer Size for Different CWT Methods.

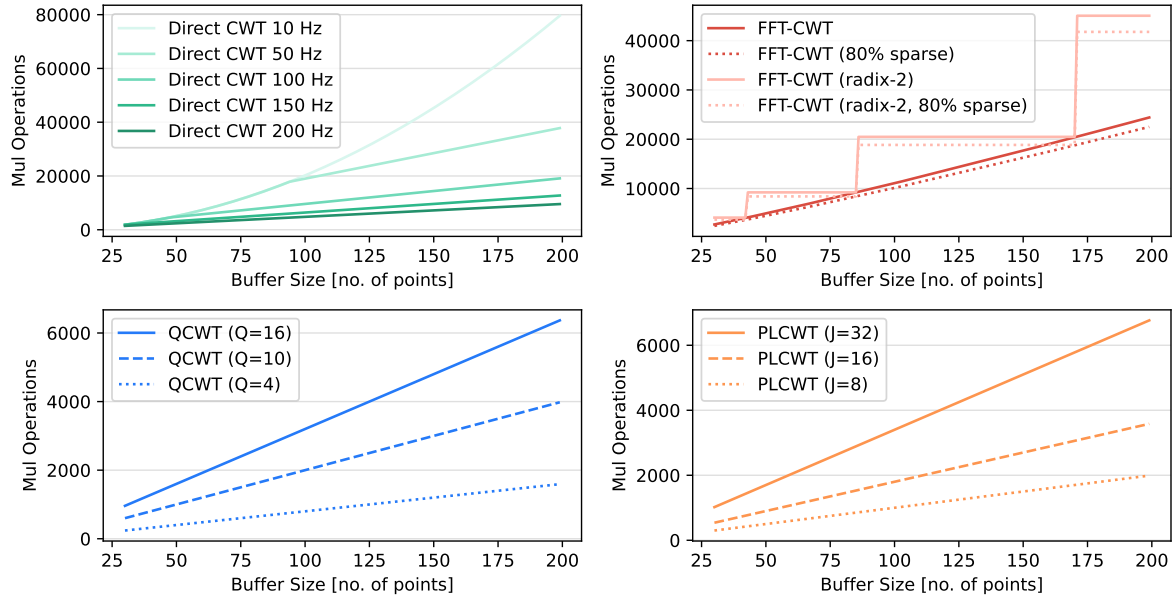


Figure 3.8: Number of Multiplications as a Function of Buffer Size for Different CWT Methods.

For the CWT and the QCWT, the number of additions increases with the buffer size S. At low frequencies, the wavelet length N also grows with S, leading to a quadratic profile. At higher frequencies, N is limited by the shorter wavelet support, so the cost only increases linearly with S. For multiplications, a similar trend is observed for the Direct CWT, with quadratic profile at low frequencies and linear at higher frequencies. However, for the QCWT, the number of multiplications depends on the quantization level Q instead of the wavelet length N, resulting in a linear profile with respect to S.

In the FFT-based CWT, the number of arithmetic operations depends on the transform length M. With a radix-2 implementation, the FFT length is chosen as the next power-of-two, which yields a staircase profile for the number of arithmetic operations. In the ideal case where non power-of-two lengths could be matched to the radix-2 efficiency, the staircase effect would disappear, yielding a nearly-linear profile with the buffer size S. Sparsity reduces the number of frequency-domain complex multiplications, hence the number of real multiplications and additions. This reduction remains however limited, as the number of operations seems to be mainly determined by the FFT and iFFT steps.

For the PLCWT, the number of additions and multiplications scales linearly with the buffer size S. In fact, for each convolution point, the number of operations is bounded by the subdivision size J. Similarly to the FFT-CWT, the number of operations does not depend on the extracted frequency.

Overall, for a given buffer size S, both QCWT and PLCWT require fewer operations than the exact methods. The QCWT achieves a $\approx 5\times$ reduction in multiplications compared to the FFT-based CWT and a $\approx 10\times$ reduction compared to the Direct CWT. The PLCWT yields similar reduction for both additions and multiplications. The exact operation savings depend on Q and J, as will be discussed in Section 3.5.

Conclusion 3.3.3.1

The QCWT reduces the number of multiplications, while the PLCWT reduces both additions and multiplications. The exact savings depend on the chosen trade-off between accuracy and the level of approximation.

3.3.3.2 Impact of CWT Parameters on Energy Consumption

The previous section analyzed the arithmetic complexity of the four methods. We now use these operation counts to estimate the energy consumption using the reference per-operation energy costs in Table 3.2.

Fixed-Point Energy

Figure 3.9 reports the energy consumption when using 32-bit fixed-point operations. For most methods (Direct CWT, FFT-CWT and PLCWT) the number of additions is comparable to the number of multiplications. For those, the energy profiles are similar to the multiplication curves, since multiplications are $31\times$ more expensive than additions (3.1 pJ vs 0.1 pJ). However, in the case of the QCWT, the number of multiplications grows only linearly with S, while additions scale quadratically. As a result, for some buffer sizes ($\geq$ 100 ms), additions can outnumber multiplications by an order of magnitude, and their contribution to the total energy thus becomes non-negligible despite the lower per unit cost. Overall, approximate methods are able to achieve lower energy than exact methods, highlighting the benefit of reducing the number of operations.

Floating-Point Energy

Figure 3.10 shows the energy estimation when using 32-bit floating-point arithmetic precision. In this case, the energy per multiplication is only $4\times$ higher than that of an addition (3.7 pJ vs 0.9 pJ). Consequently, additions contribute more to the total cost than in the fixed-point case. For the same configuration, the floating-point energy is higher than that for fixed-point, typically by more than 50%.

For single-frequency extraction, the Direct CWT and the FFT-CWT are comparable: the Direct CWT is more efficient at high frequencies due to shorter wavelet support, while

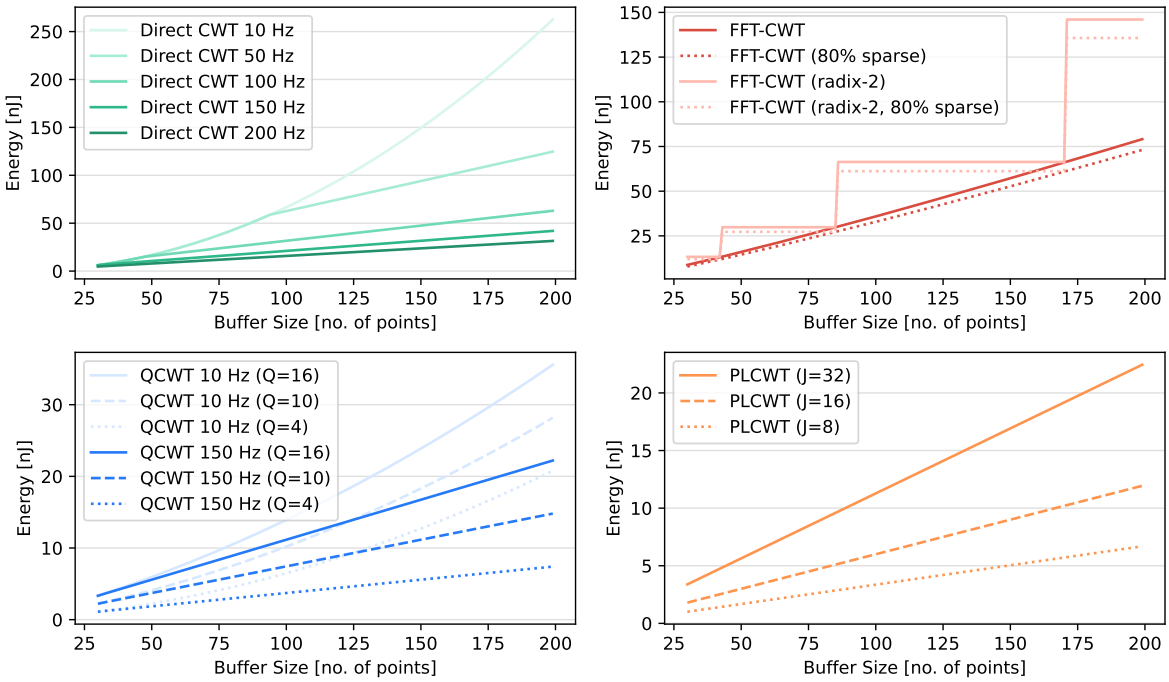


Figure 3.9: Energy for a Single CWT Operation Using Fixed-Point Arithmetic, as a Function of Buffer Size.

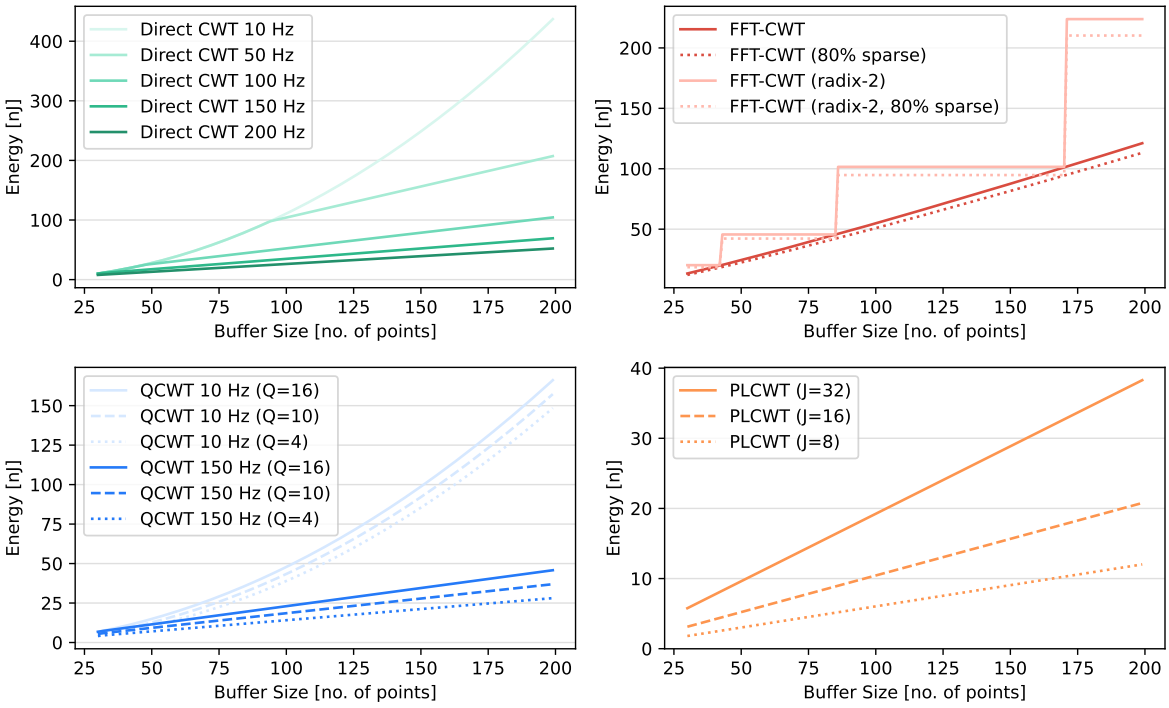


Figure 3.10: Energy for a Single CWT Operation Using Floating-Point Arithmetic, as a Function of Buffer Size.

the FFT-CWT requires less energy for lower frequencies. The QCWT's benefit is limited at low frequencies where the number of additions dominates. In contrast, the PLCWT remains efficient since both additions and multiplications are bounded by the subdivision size J, which makes it attractive at low-frequency bands, as it avoids the quadratic cost explosion observed in the QCWT.

Conclusion 3.3.3.1

In fixed-point precision, energy is mainly driven by multiplications, while in floating-point, additions also contribute. The QCWT and PLCWT achieve significant savings by reducing the number of arithmetic operations compared to the exact methods. This reduction is, however, limited when using the QCWT to extract low frequency components in floating-point precision.

3.3.3.3 Multi-band Feature Extraction

Many BCI systems extract features across multiple frequency bands. We quantify how the number of frequencies F affects the mean energy per extracted band. We fix the buffer size to $S = 100$ ms and vary $F \in [1, 20]$ to extract the components $\{\, i \times 10 \text{ Hz} \mid i = 1, \ldots, F \,\}$. Note that, for all methods, the total energy increases with F as more bands are processed. Figure 3.11 reports the **per-band energy** for all methods in fixed-point precision.

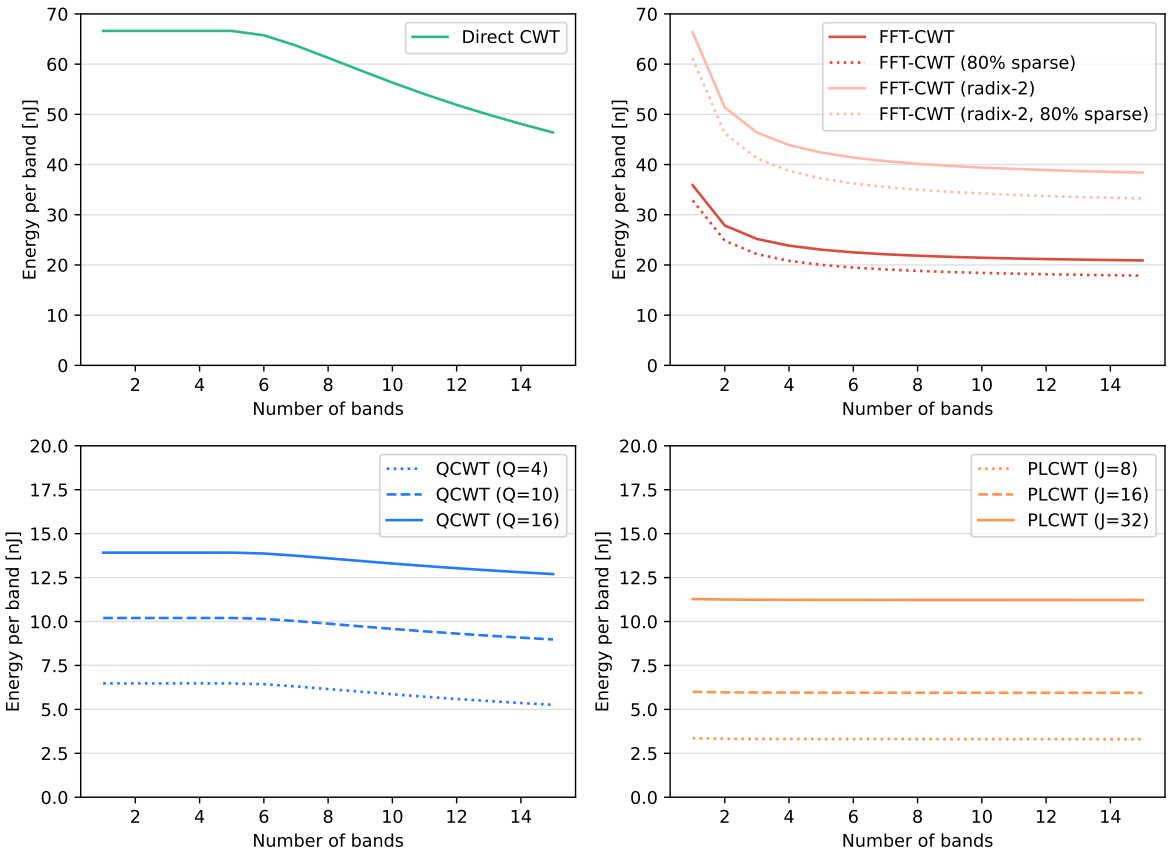


Figure 3.11: Mean Per-Band Energy as a Function of the Number of Extracted Bands F at 100 ms for All CWT Methods

For the Direct CWT, the per-band energy decreases with each higher frequency that

we add, as these frequencies have shorter wavelet support and therefore lower compute cost. For the FFT-CWT, the per-band energy decreses with F then reaches a plateau. This reflects the amortization of the FFT cost over a larger number of bands. The radix-2 implementation has a higher cost than the theoretical limit due to padding overhead. Sparsity optimization reduces the frequency-domain multiplications, however, the iFFT step still dominates the extraction cost. For QCWT, the per-band energy is determined by the number of quantization levels Q, with a slightly decreasing trend for higher F values, due to the higher frequencies. Similarly, the PLCWT presents a nearly flat per-band energy versus F curve. This energy consumption depends on the subdivision parameter J. Although the integration stage introduces a fixed overhead per buffer that is amortized across multiple bands, its energy cost remains negligible compared to the reduction step, and therefore does not significantly affect the trends.

Conclusion 3.3.3.3

Approximate methods have a lower per-band energy that depends on the approximation parameters (Q for QCWT, J for PLCWT) rather than on the extracted frequency. As a result, even for a single band (F=1) extraction, they can be more efficient than exact methods, including FFT-CWT after amortization. Different bands may require different Q or J values to reach a target quality, which will be examined in 3.5.1.

3.4 BCI Datasets, Decoder and Evaluation Metrics

To evaluate the proposed approximate CWT methods, we consider real BCI scenarios. This section describes the datasets, the decoder, and the metrics used to evaluate feature quality and decoding accuracy. In addition, we compute the computational and energy cost based on the operation and energy estimation in Section 3.3.

3.4.1 Datasets

We evaluate the proposed methods on two Electrocorticography (ECoG) datasets. The first is from the BCI Competition IV focused on finger movements [143], while the second is a clinical dataset where a tetraplegic patient controlled a virtual avatar and an exoskeleton with multiple Degrees of Freedom (DoFs) [20]. These setups cover experimental recordings and real-life application scenarios.

3.4.1.1 BCI Competition IV Dataset

The first dataset [143] corresponds to finger movement tasks from the BCI Competition IV. ECoG signals were recorded from three epileptic subjects while performing individual finger movements. Signal were sampled at 1 kHz using electrode grids placed on the sensorimotor cortex, with 48 to 64 channels depending on the subject. Each finger was equipped with a flexion sensor, providing continuous movement traces aligned with the ECoG recordings. The position sampling rate is 25 Hz, corresponding to non-overlapping windows of 40 ms. For our application, the features are however computed over sliding windows of 80 ms. We retained data from subjects **S1** and **S3**, as the recordings of subject **S2** showed high magnitude artefacts due to epileptic seizure. Finger positions were

thresholded to define discrete states, as previously done in [5]. We define two classification tasks for BCI decoding:

- a **2-class** problem distinguishing between movement and rest,
- a **5-class** problem classifying movements of individual fingers while excluding the idle state.

This dataset is widely used for benchmarking motor BCIs [46, 87, 173], providing a test to assess feature extraction and decoding methods.

3.4.1.2 8-DoF Avatar Control Dataset

The second dataset is derived from a clinical proof-of-concept study in which a tetraplegic patient controlled a virtual avatar and a four-limb exoskeleton using an ECoG-based system [20]. Two arrays, each with 64 electrodes, were implanted over the sensorimotor cortex to record ECoG signals at 590 Hz. Only 32 channels per implant were streamed due to wireless bandwidth limitations.

Motor imagery was decoded with a Markov Switching Linear Model (MSLM), trained online using a Recursive Exponentially Weighted N-way Partial Least Squares (REW-NPLS). Model updates were performed every 15 s, while control commands were issued to the effector at 10 Hz. For our tests, we focus on the avatar control experiments, and more specifically, on a five-state classification between the following: idle, left and right hand, and left and right wrist. This dataset illustrates a clinical scenario of BCI motor control, providing a benchmark to evaluate the efficiency of approximate features.

3.4.2 Decoder

For both datasets, decoding was performed with a MSLM that models the temporal dynamics of brain activity while taking into account state transitions through a Hidden Markov Model (HMM). This approach has been shown to provide robust performance in asynchronous BCI [142]. The model parameters were trained online using a REW-NPLS algorithm, which incrementally updates the decoding model. In our setup, features were computed over sliding windows of 80 ms for the BCI Competition IV dataset, and 100 ms for the avatar control dataset. For the first dataset, we extract 20 frequency bands, equally spaced, between 10 Hz and 200 Hz, and for the second one we use the same features as in [20], which consist of frequency bands between 10 Hz and 150 Hz. For the decoding step, we use ten time windows for all classification tasks, which corresponds to 800 ms for the 2-class and 5-class BCI IV tasks, and to 1 s for the 8-DoF database.

3.4.3 Evaluation Metrics

To assess the proposed approximate CWT methods, we evaluate the cost, feature quality and decoding performance. First, we compute the energy requirements derived from the estimation models in Section 3.3. We then evaluate the fidelity of the extracted features by measuring their correlation with exact CWT features. Finally, we report the decoding performance across the tested databases and tasks.

3.4.3.1 Energy Comparison of Approximate Methods

We analyze how the total energy consumption per signal channel evolves as a function of the approximation parameters Q (for the QCWT) and J (for the PLCWT).

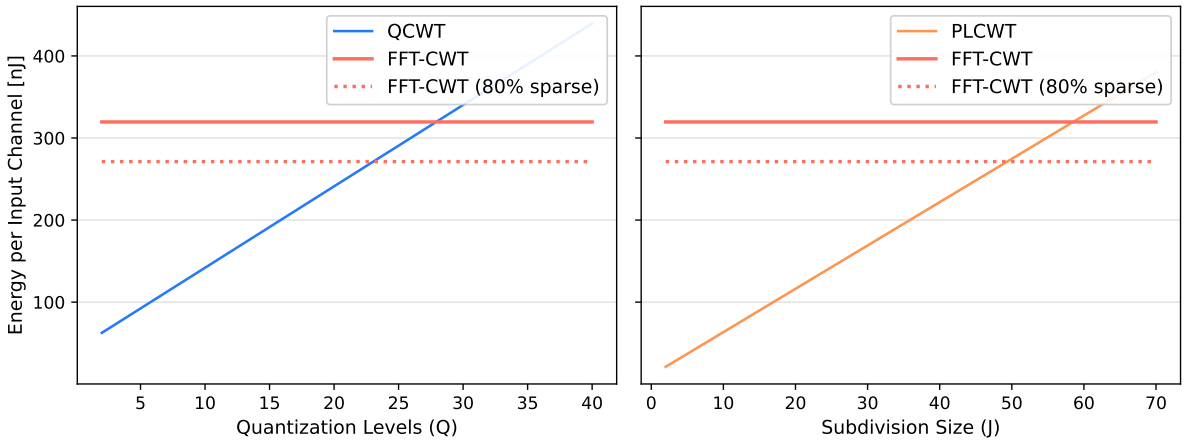


Figure 3.12: Energy per Signal Channel as a Function of Q (for QCWT) and J (for PLCWT) for the BCI Competition IV Dataset Using 32-Bit Fixed-Point Precision.

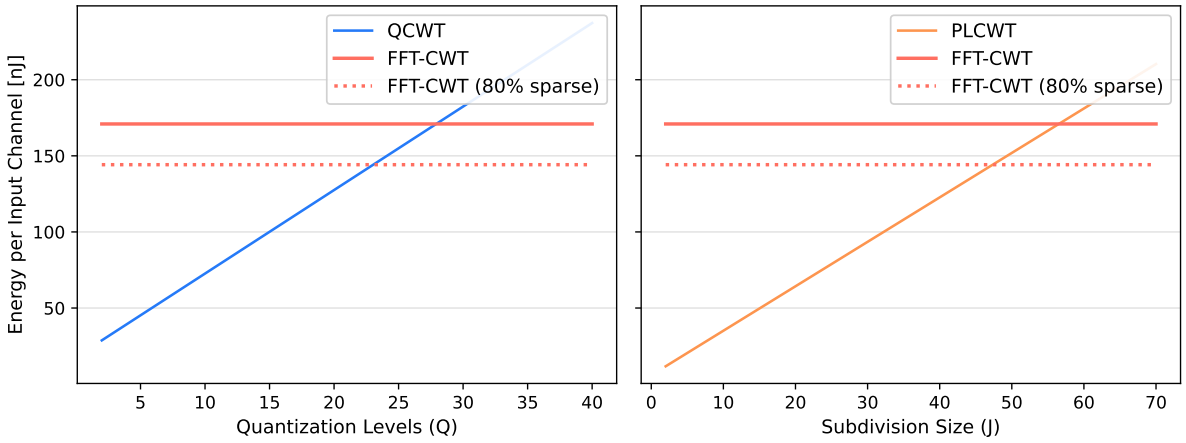


Figure 3.13: Energy per Signal Channel as a Function of Q (for QCWT) and J (for PLCWT) for the 8-DoF Avatar Control Dataset Using 32-Bit Fixed-Point Precision.

The energy consumption was estimated using the models in Section 3.3 and the 32-bit fixed-point reference values of Table 3.2. All results are reported for a single electrode, as the total cost in a BCI system scales linearly with the number of channels.

For the BCI Competition IV dataset, results are shown in Figure 3.12. The QCWT becomes more efficient than the sparse FFT-CWT for $Q \leq 22$, and achieves a 3× reduction at $Q = 5$. The PLCWT requires less energy than the FFT baselines for $J \leq 50$, reaching up to a 4× reduction at $J = 12$.

In Figure 3.13 the same variations are plotted for the 8-DoF avatar dataset. Due to the lower sampling frequency and smaller number of bands, the FFT-CWT cost is reduced. However, the same overall trend is observed and both approximate methods achieve significant energy savings for low values of Q and J.

Conclusion 3.4.3.1

Approximate methods reduce the energy cost by up to 4× compared to a FFT-based CWT in a real BCI system, depending on the selected approximation parameters Q and J.

3.4.3.2 Feature Correlation

We evaluate the similarity between exact and approximate features. Since the output of the CWT is complex-valued, we measure the Correlation Coefficient (CC) for the real and imaginary parts between the exact and approximate outputs. In addition, we compute the correlation of the L_2-norm features, before and after applying the linear correction described in Section 3.3. This analysis is carried out for different approximation parameters Q and J. For each frequency, we report the average CC across channels, which provides a metric to assess the approximate CWT at a given frequency. A high CC indicates that the approximate features retain most of the information contained in the exact features. When the CC is sufficiently close to 1, the approximate features yield comparable decoding performance, as it will be shown in Section 3.5.

3.4.3.3 Decoding Performance

Finally, we evaluate the impact of approximate features on decoding performance. Decoding performance is quantified using standard classification metrics [154]: accuracy, precision, recall, and the F1-score.

TP, TN, FP, and FN denote the number of true positives, true negatives, false positives, and false negatives respectively. The metrics are defined as [40]:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.16}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.17}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.18}$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.19}$$

For multi-class problems with C classes, we compute each metric per class in a one-vs-rest manner and then take the unweighted mean across classes (macro-averaging). For the recall metric, the mean value is called the balanced accuracy given by:

$$\text{Balanced Accuracy} = \frac{1}{C} \sum_{c=1}^{C} \frac{TP_c}{TP_c + FN_c} \tag{3.20}$$

3.5 Results and Discussion

In this section, we present the evaluation of the proposed approximate CWT methods on the two ECoG datasets introduced in Section 3.4. We first analyze the correlation between exact and approximate features and then assess the decoding performance. Finally, we link these results to the computational models of Section 3.3, to highlight the trade-offs between algorithmic performance and energy consumption.

3.5.1 Feature Correlation for Approximate CWT

We first evaluate how the approximate methods (QCWT and PLCWT) preserve the information contained in the exact CWT. We distinguish between two cases: the raw

real and imaginary parts of the CWT outputs, and the L_2-norm features, before and after linear correction. All results are obtained using ECoG recordings from the BCI Competition IV dataset.

3.5.1.1 Raw Complex Correlation

For each frequency, the similarity is quantified using the CC between exact and approximate features, as detailed in 3.4.3.2. We analyze how the approximation parameters Q and J affect the CC.

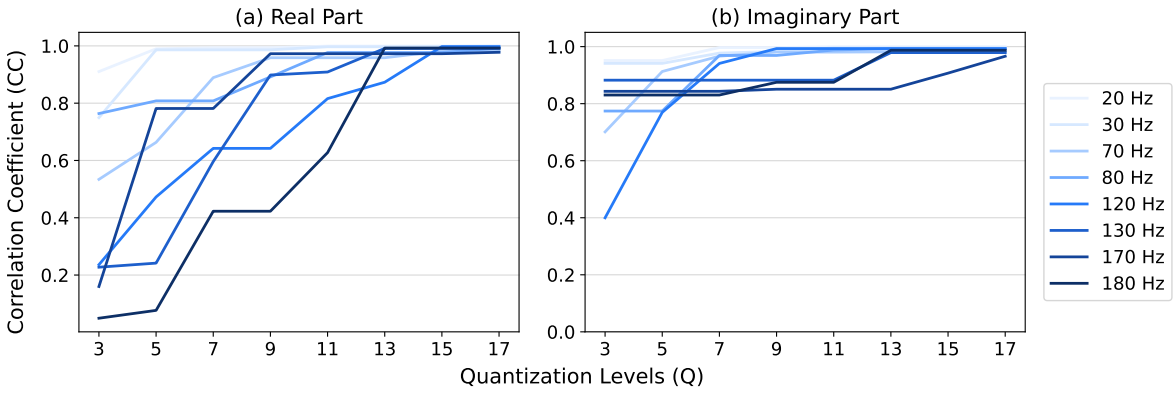


Figure 3.14: Correlation Coefficient for the (a) Real and (b) Imaginary Parts of the Outputs as a Function of Quantization Levels Q for the QCWT.

Figure 3.14 shows the CC for the real and imaginary parts of the QCWT output for different frequency bands, as a function of the number of quantization levels Q. As expected, the CC increases with Q, as more quantization levels give a better approximation of the wavelet. Low-frequency bands reach CC values close to 1 more rapidly than high-frequency bands. In fact, low-frequency wavelets have fewer oscillations, which makes them easier to approximate. For instance, low frequencies (20-30 Hz) exceed a CC of 0.95 for small Q values (≈ 5), while high frequencies (170-180 Hz) require larger Q (≈ 13) to reach the same threshold. The imaginary part shows higher CC at low Q, which can be explained by the antisymmetry of the Morlet wavelet. As seen in 3.2.1.3, each non-zero quantization value v_q corresponds to both $+v_q$ and $-v_q$. For antisymmetric wavelets, when an optimal quantization value is selected, the opposite value is automatically relevant due to the wavelet antisymmetry. In contrast, for symmetric wavelets, the opposite sign is not necessarily useful.

Figure 3.15 reports the CC between exact and approximate features for the PLCWT, as a function of the subdivision size J. The correlation increases with J, since larger subdivisions provide a better approximation of the wavelet. Similarly to the QCWT, low-frequency components reach high CC values for a lower approximation parameter than high-frequency ones. For example, for frequencies of 20-30 Hz the CCs exceed 0.95 for $J \approx 10$, while higher frequency bands (170-180 Hz) require larger J values (≈ 35) to achieve the same level of correlation. Real and imaginary parts follow nearly identical trends while varying J.

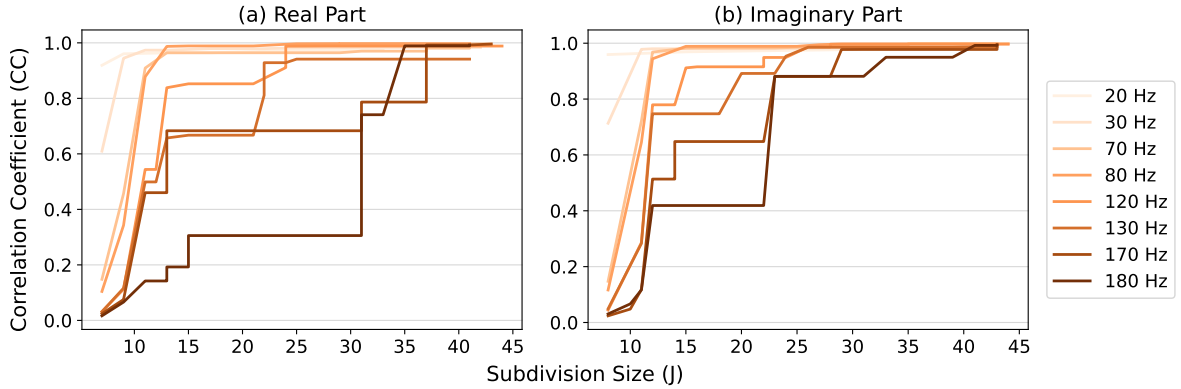


Figure 3.15: Correlation Coefficient for the (a) Real and (b) Imaginary Parts of the Outputs as a Function of Subdivision Size J for the PLCWT.

Conclusion 3.5.1

Low-frequency components are easier to approximate and reach high CC values with small approximation parameters (Q and J), while higher frequencies require larger parameters to reach the same correlation. Nevertheless, approximate methods can preserve high feature fidelity depending on the parameter values.

3.5.1.2 L_2-Norm Correlation

Each approximation parameter value (Q and J) can be directly mapped to an energy cost using the models in Section 3.3. This allows us to evaluate the trade-off between feature correlation and energy, and to identify Pareto-optimal configurations where no further gain in correlation can be achieved without increasing energy.

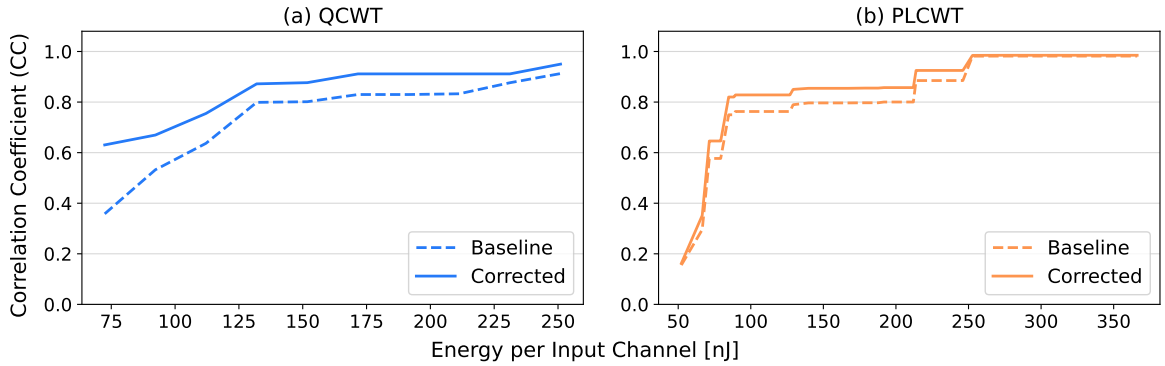


Figure 3.16: Correlation Coefficient of the L_2-Norm Features as a Function of Total Energy for the (a) QCWT and (b) PLCWT, Before and After Applying the Linear Correction.

Figure 3.16 shows the CC of the L_2-norm features as a function of the total energy per input channel. For both methods, the CC increases, as expected, with the energy consumption, and applying the linear correction shifts the curves upward, which proves the method efficiency. Although the correction step appears more effective for the QCWT, both methods provide comparable Pareto-optimal solutions. We use the corrected features in the remainder of this work.

Conclusion 3.5.1.2

When increasing the energy budget, a higher level of feature correlation can be reached. Moreover, the proposed linear correction improves feature correlation at a constant energy consumption.

3.5.2 Decoding Performance

We now assess how feature correlation translates into decoding performance. For reference, we first report the decoding accuracy obtained in the different datasets using the exact CWT features. For the BCI Competition IV dataset, the MSLM decoder achieves an accuracy of 68% (resp. 66%) for the 2-class (resp. 5-class) task, with subject S1's data. Similar accuracy results (73% resp. 69%) are obtained for subject S3. The same decoder, trained and tested on the 8-DoF database, achieves an average accuracy of 76%. Note, the HMM transition matrix is only applied to the second dataset, where transitions between states are limited when collecting the data.

3.5.2.1 Balanced Accuracy vs. Correlation

For each approximate method, we compute the balanced accuracy loss (relative to the exact CWT baseline) as a function of the average CC of the approximate features across all channels and frequencies.

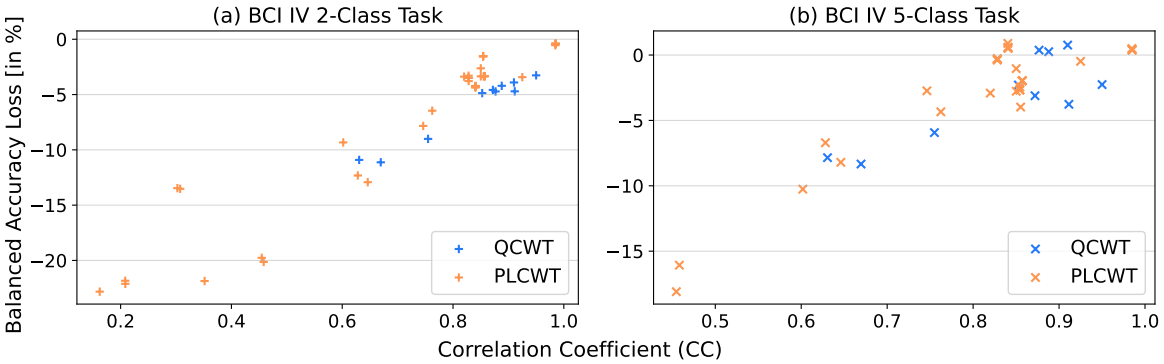


Figure 3.17: Balanced Accuracy Loss vs. CC for the (a) 2-Class and (b) 5-Class BCI Competition IV Tasks. Each Point Corresponds to a Wavelet Approximation.

The results for the BCI Competition IV dataset are shown in Figure 3.17. Balanced accuracy loss decreases when CC increases for both tasks. For the 2-class task, losses remain within 5% once the average CC exceeds 0.8, indicating a correlation threshold above which the accuracy loss is limited. For the 5-class task, the same trend holds and the same CC threshold is sufficient to limit the loss below 5%.

Results for the 8-DoF avatar dataset are reported in Figure 3.18. Here, the decoder appears to be less sensitive to approximation, with losses remaining under 5% even when the average CC is as low as 0.5. This robustness can be explained by the state transition matrix, which constrains the set of valid transitions and, hence, reduces the effect of feature degradation on the output classification.

Overall, for all tasks, high correlation values consistently limit accuracy loss. These results suggest that the CC provides a practical way of selecting approximation parameters Q and J that balance energy consumption and decoding accuracy.

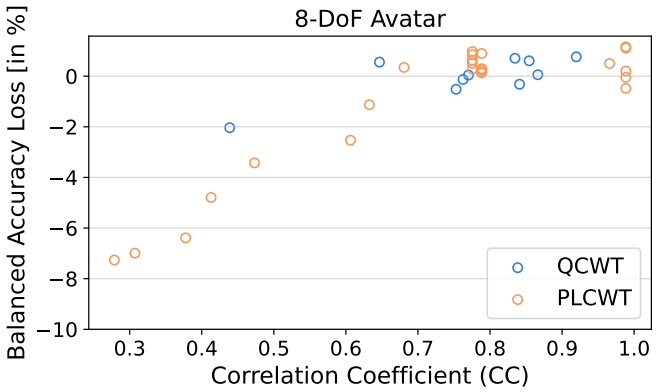


Figure 3.18: Balanced Accuracy Loss vs. CC for the 8-DoF Avatar Control Dataset.

Conclusion 3.5.2.1

The trend between CC and balanced accuracy shows that correlation can serve as a proxy to evaluate decoding performance, enabling the validation of approximate features before full BCI experiments.

3.5.2.2 Correlation-Based Configurations

Based on the CC vs. accuracy analysis, we can fix a threshold for the average CC required for a given decoding accuracy. This average value can be reached by fixing the same threshold for all frequency bands, or by leveraging CC profiles that achieve energy trade-offs. We evaluate three strategies for choosing the approximation parameters to achieve this average CC.

- **Constant (C):** the CC at each frequency band is constrained to remain above a fixed threshold.
- **Decreasing (D):** a linearly decreasing threshold profile is applied, starting at 0.95 for the lowest frequency and gradually decreasing for higher frequencies. This reduces the energy cost by sacrificing the quality of high frequency features, while maintaining a good CC for the low frequencies to obtain the desired mean correlation.
- **Constant-Decreasing (CD):** the CC is constrained to remain constant up to a given frequency, after which a decreasing threshold profile is applied. This hybrid strategy combines the constant profile at low frequencies with the decreasing profile at higher frequencies for energy reduction.

3.5.2.3 Classification Metrics

To evaluate decoding performance under the different approximation profiles, we report the classification metrics defined in 3.4.3.3. For multi-class tasks, metrics are macro-averaged across classes.

3.5.2.4 BCI Competition IV Dataset

Tables 3.3 and 3.4 summarize the classification metrics obtained for the 2-class and 5-class tasks of the BCI Competition IV dataset. The decoding is evaluated using the FFT-CWT exact features, and the three correlation-based configurations defined for each

approximate CWT method. Note, the previously defined balanced accuracy corresponds to the recall.

Table 3.3: BCI Competition IV, 2-Class Task Classification Metrics (in %).

Method	Subject S1				Subject S3			
	Accuracy	Recall	Precision	F1-Score	Accuracy	Recall	Precision	F1-Score
FFT (baseline)	71.7	67.9	68.9	68.3	**72.6**	**73.3**	**75.6**	**72.2**
QCWT-C	73.8	69.1	71.5	69.8	64.2	65.4	70.2	62.3
QCWT-D	71.2	66.2	68.3	66.8	65.2	66.4	71.7	63.4
QCWT-CD	72.4	66.5	70.0	67.3	70.3	71.0	72.9	69.9
PLCWT-C	**74.4**	**70.1**	**72.1**	**70.8**	70.8	71.5	73.2	70.4
PLCWT-D	73.5	68.3	71.3	69.1	65.4	66.5	71.5	63.6
PLCWT-CD	72.4	66.8	70.0	67.6	65.6	66.6	71.1	64.0

For the 2-class task (Table 3.3), the FFT-CWT baseline achieves 71.7% accuracy for **S1** and 72.6% for **S3**. Both approximate methods preserve this level, with PLCWT-C slightly outperforming the baseline on **S1**. For **S3**, approximation has more impact on the decoding performance. The metrics degrade across many approximation profiles, especially the decreasing (D) one for both methods. The QCWT-CD and the PLCWT-C configurations are able to preserve performance close to the baseline. These observations highlight the importance of the different approximation strategies. Depending on the subject and task, a constant, decreasing, or hybrid profile may better balance energy reduction with classification performance.

Table 3.4: BCI Competition IV, 5-Class Task Classification Metrics (in %).

Method	Subject S1				Subject S3			
	Accuracy	Recall	Precision	F1-Score	Accuracy	Recall	Precision	F1-Score
FFT-CWT	65.6	66.1	**68.3**	63.2	**68.3**	68.7	**67.2**	**67.1**
QCWT-C	61.4	60.9	58.5	53.7	66.4	68.2	66.2	66.0
QCWT-D	62.5	62.2	57.9	55.4	54.2	55.9	60.2	53.4
QCWT-CD	**67.3**	**66.6**	66.9	**64.5**	52.3	54.2	58.4	52.0
PLCWT-C	64.6	64.7	67.8	63.0	67.4	**68.9**	66.7	66.7
PLCWT-D	65.2	65.0	67.4	61.9	67.8	68.3	66.6	66.8
PLCWT-CD	65.8	65.8	67.6	63.1	54.5	56.7	60.4	53.5

For the 5-class task (Table 3.4), the FFT-CWT baseline achieves 65.6% accuracy for **S1** and 68.3% for **S3**, significantly higher than the chance level (20%). For **S1**, all PLCWT profiles remain close to the baseline, while the QCWT results are less stable, with only the QCWT-CD slightly outperforming the baseline metrics. For **S3**, performance is more sensitive to approximation. The PLCWT-C and PLCWT-D configurations remain close to the baseline, while QCWT-C also provides stable results. Overall, despite the heterogeneous performance of the configurations across subjects and tasks, the PLCWT-C configuration consistently shows the most stable decoding, while other profiles can still provide good results in some cases.

To further illustrate these trends, we report the confusion matrices for **S3** in the 5-class task when using the constant (C) (Figure 3.19) and constant-decreasing (CD) (Figure 3.20) configurations. We compare these approximate setups to the FFT baseline.

As shown in Figure 3.19, both QCWT-C and PLCWT-C preserve the baseline structure, with predictions concentrated along the diagonal. In contrast, Figure 3.20 highlights

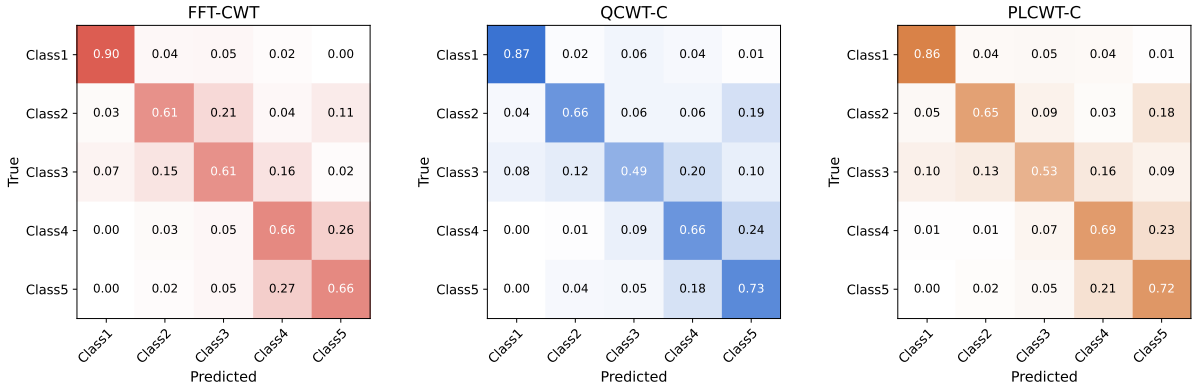


Figure 3.19: Confusion Matrices for the BCI Competition 5-class Task (**S3** Dataset) Using FFT Baseline, QCWT-C and PLCWT-C Configurations.

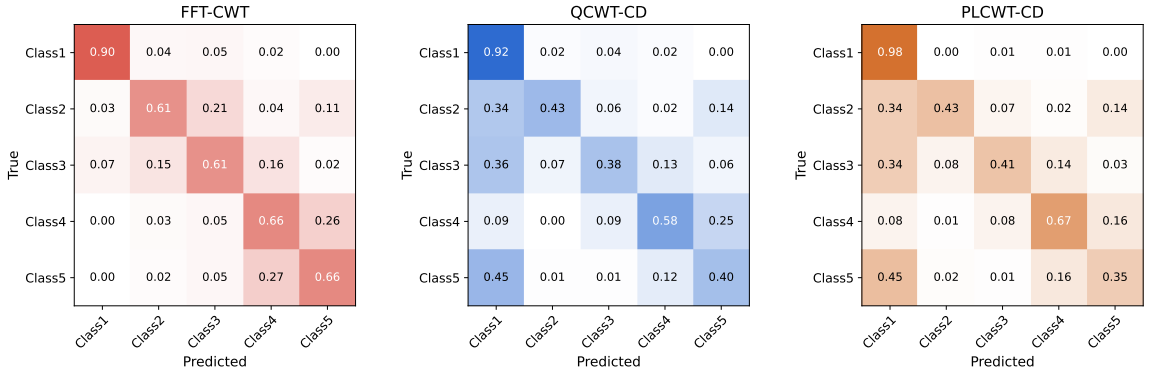


Figure 3.20: Confusion Matrices for the BCI Competition 5-class Task (**S3** Dataset) Using FFT-CWT, QCWT-CD and PLCWT-CD Configurations.

how the (CD) profile impacts decoding. In fact, the QCWT-D and the PLCWT-D configurations introduce prediction errors (off-diagonal), which can be attributed to feature degradation caused by the more aggressive approximation of high frequency components.

Conclusion 3.5.2.4

For the BCI Competition IV dataset, the PLCWT-C configuration shows the most stable decoding across tasks and subjects. Other configurations provide good performance in some cases and may offer interesting energy trade-offs.

3.5.2.5 8-DoF Avatar Dataset

Table 3.5 reports the classification metrics averaged over the 32 sessions of the 8-DoF avatar dataset, when using the FFT-CWT and the QCWT and PLCWT approximate methods. All approximate configurations preserve or slightly improve performance across these metrics. The best mean performance is obtained by PLCWT-D although the PLCWT-C yields close performance. Overall, the 8-DoF decoder performnce appears more stable, which can be explainde by the constrained transition matrix.

Figure 3.21 shows, in details, the distribution of these metrics across sessions. The boxplots show tight distributions with small variations in medians and interquartile ranges,

Table 3.5: Avatar 8-DoF: Classification Metrics (Mean across Sessions, in %).

Method	Accuracy	Recall	Precision	F1-Score
FFT-CWT	79.3	76.8	74.7	72.0
QCWT-C	80.6	77.3	74.9	73.1
QCWT-D	80.9	77.6	75.2	73.5
QCWT-CD	80.7	77.4	75.5	73.4
PLCWT-C	81.3	78.0	75.7	74.0
PLCWT-D	**81.9**	**78.4**	**76.3**	**74.6**
PLCWT-CD	80.5	77.8	75.4	73.4

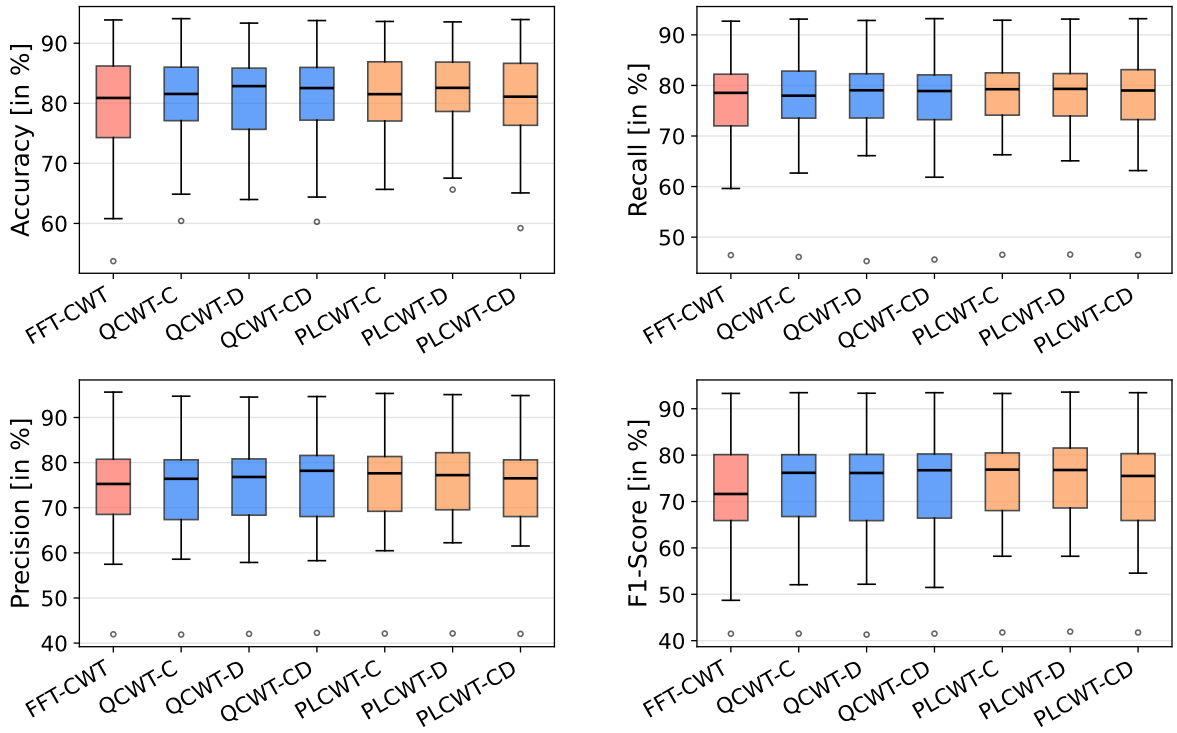


Figure 3.21: Distributions of Classification Metrics across Sessions for the 8-DoF Avatar Dataset Using FFT-CWT, QCWT, and PLCWT Configurations.

indicating stable performance. The QCWT and PLCWT performances are comparable for all correlation profiles. Unlike the BCI Competition IV case, the decreasing (D) profiles do not degrade performance and can even yield small improvements.

Conclusion 3.5.2.5

For the 8-DoF avatar dataset, all QCWT and PLCWT configurations match baseline decoding performance, due to the constrained state transitions.

3.5.3 Performance-Energy Trade-offs

We combine decoding performance and energy cost by using the analytical per-channel energy model of Section 3.3 and the decoding performance of different configurations in 3.5.2. Balanced accuracy (mean recall) is reported as a baseline value for the FFT-CWT across all datasets, with the corresponding per-channel energy consumption in nJ. Baseline

energies are 320 nJ for BCI IV and 171 nJ for 8-DoF. We compare the approximate configurations by reporting the accuracy loss in percent, with the energy reduction factor achieved. Results are shown in Table 3.6.

Table 3.6: Summary of the Balanced Accuracy Loss and Energy Reduction Factor of the Approximate Methods Compared to the FFT-CWT Baseline Accuracy and Energy Cost.

Method	**Balanced Accuracy / Loss**					**Energy / Reduction**	
	2-Class S1	**2-Class S3**	**5-Class S1**	**5-Class S3**	**8-DoF**	**BCI-IV**	**8-DoF**
FFT-CWT	67.9%	73.3%	66.1%	68.7%	76.8%	320 nJ	171 nJ
QCWT-C	+ 1.2%	− 7.9%	− 5.2%	− 0.5%	+ 0.5%	3×	4×
QCWT-D	− 1.7%	− 6.9%	− 3.9%	− 12.8%	+ 0.8%	3×	4×
QCWT-CD	− 1.4%	− 2.3%	**+ 0.5%**	− 14.5%	+ 0.6%	3×	4×
PLCWT-C	**+ 2.2%**	**− 1.8%**	− 1.4%	**+ 0.2%**	+ 1.2%	3×	4×
PLCWT-D	+ 0.4%	− 6.8%	− 1.1%	− 0.4%	**+ 1.6%**	4×	4×
PLCWT-CD	− 1.1%	− 6.7%	− 0.3%	− 12.0%	+ 1.0%	4×	4×

Overall, **PLCWT-C** offers the most stable trade-off on both datasets while reducing energy by a factor of 3× on the BCI-IV data and 4× on the 8-DoF database. Other configurations can also be interesting in specific cases such as the PLCWT-D on BCI IV 2-class and 5-class tasks for subject **S3**, achieving a 4× reduction in energy consumption. On the 8-DoF dataset, all approximate configurations match or slightly improve the baseline, while maintaining a 4× energy reduction compared to the FFT-CWT baseline.

Conclusion 3.5.3

QCWT and PLCWT achieve 3× to 4× lower energy consumption with less than 2% balanced-accuracy loss on both datasets. Despite varying performance of the configurations, the PLCWT-C maintains stable accuracy across tasks and subjects.

3.6 Conclusion

In this chapter, we introduced two approximate CWT methods (QCWT and PLCWT) and validated them with real ECoG datasets. We showed that low-frequency components were easier to approximate than high-frequency bands. A simple linear post-correction improved the CC at constant energy, yielding a better Pareto front. We linked feature correlation to decoding performance.

Furthermore, we quantified energy savings using energy cost models. Relative to the FFT-CWT baseline, the proposed approximations achieved 3-4× lower energy while preserving high decoding accuracy, with losses $\leq 2\%$ compared to the exact CWT baseline. Overall, the results demonstrated that approximate wavelet transforms can enable energy reduction for BCI feature extraction with virtually no impact on decoding accuracy. Finally, the methods provided a tunable parameter (Q for QCWT and J for PLCWT) that fixes the trade-off between energy consumption and approximation qualtiy.

Chapter 4

An Adaptive Hardware Accelerator for Wavelet-Based Feature Extraction

Chapter 3 introduced approximate methods to compute the Continuous Wavelet Transform (CWT), showing that piecewise approximations can significantly reduce compute complexity while preserving decoding performance. In addition, energy models were used to estimate the expected power savings at the algorithmic level, but these remain to be validated for an actual hardware implementation. This chapter presents an Adaptive Quality Feature Extractor (AQFE), a hardware module that implements one of the approximate algorithms from the previous chapter. The circuit is based on an algorithm-to-circuit optimization of the Piecewise Linear Continuous Wavelet Transform (PLCWT) with dynamic quality control, enabling accuracy-energy trade-offs at runtime. The design is validated against a software reference model to confirm its accuracy for feature extraction, and its power consumption is analyzed through post-layout simulations and compared with state-of-the-art Brain-Computer Interface (BCI) circuits.

- Section 4.1 motivates the choice of the PLCWT technique and the multi-layer optimization strategy.
- Section 4.2 describes the architecture, including the algorithmic mapping, top-level block diagram, and architectural optimizations.
- Section 4.3 presents the physical implementation in 22 nm FDSOI technology, circuit-level optimizations, and layout views of different variants of the circuit.
- Section 4.4 reports the results of functional validation, power analysis, performance trade-offs, and compares the proposed module with state-of-the-art feature extractors.

Parts of this chapter were presented in the Design, Automation and Test in Europe Conference 2026 [138].

4.1 Motivation

The approximate CWT methods introduced in Chapter 3 showed that the PLCWT achieves significant compute reduction with limited accuracy loss. Its performance remained stable across subjects and datasets for different motor tasks, highlighting the robustness of this approach. In addition, the PLCWT provides a simple mechanism for controlling the level of approximation through the subdivision size. This flexibility gives

the feature extractor the ability to preserve accuracy during complex task decoding or reduce power when energy savings are needed, or when the task complexity is lower.

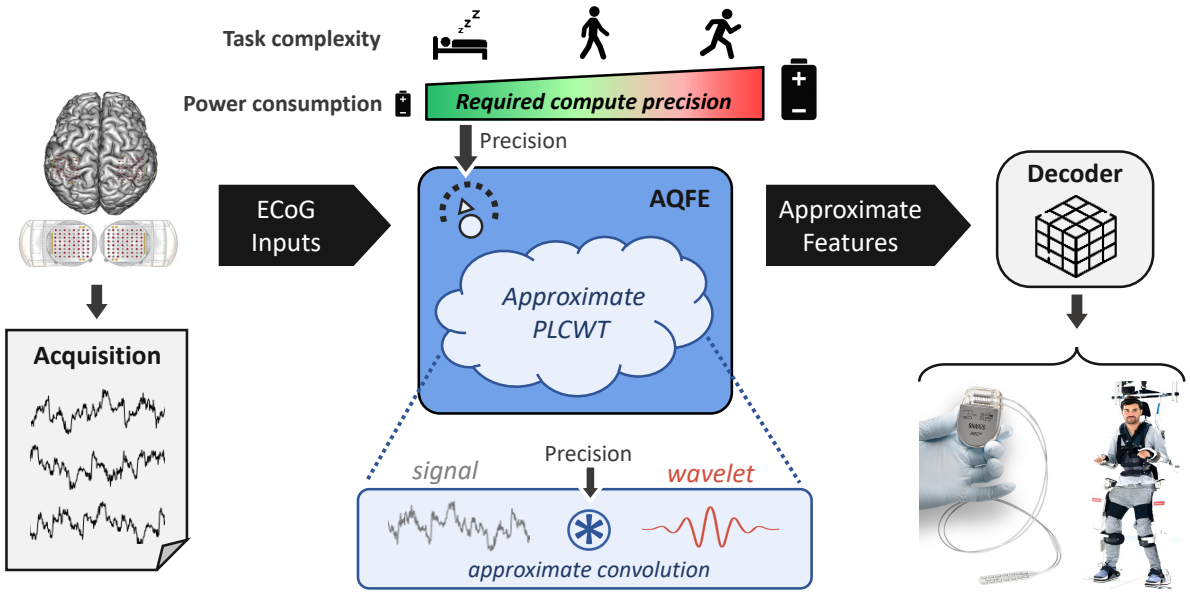


Figure 4.1: Overview of a BCI System with Adaptive-Quality PLCWT Feature Extraction

In a typical BCI use case, shown in Figure 4.1, the AQFE module extracts time-frequency features from Electrocorticography (ECoG) signals using the PLCWT with adjustable precision, while a pre-trained decoder maps the extracted features to task-specific outputs. The required precision can be selected according to task difficulty or based on external feedback, enabling adaptive operation. Implementing this AQFE in hardware leverages the algorithmic simplifications, and extends the optimization down to the circuit level.

4.1.1 Algorithmic Choice for AQFE

The PLCWT algorithm approximates the wavelet ϕ with a piecewise linear function ψ, using a limited number of linear segments defining a subdivision of its interval [135]. Examples of different approximations are illustrated in Figure 4.2.

The number of segments, referred to as the subdivision size, determines the approximation quality. Each additional subdivision point refines the wavelet approximation, while fewer segments reduce the number of multiplications per convolution, hence the energy cost.

4.1.2 Multi-Layer Optimization Approach

The proposed AQFE is designed using a top-down optimization strategy that goes from the algorithmic to the circuit layer. At the algorithmic level, simplifications reduce the number of compute operations, and subdivision control provides a mechanism to balance accuracy and energy consumption. At the architectural level, multiple parallel lane configurations and memory reuse are exploited to balance throughput, area, and energy. Finally, at the circuit level, techniques such as clock gating and process technology and cell library selection minimize dynamic and static power consumption. This multi-layer

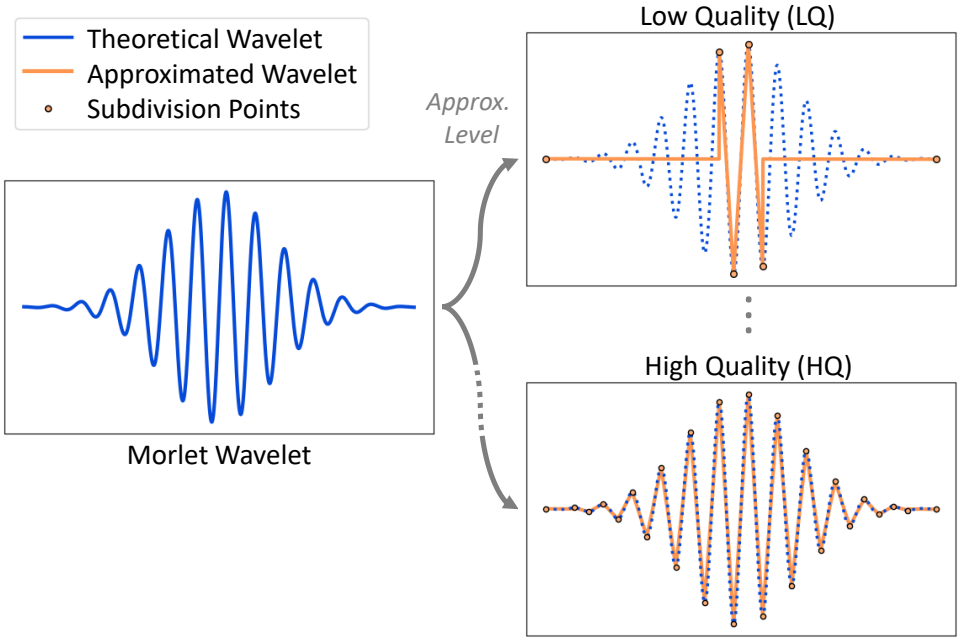


Figure 4.2: Wavelet Approximation for Different Subdivision Sizes.

approach keeps the main benefits of the PLCWT algorithm throughout the design flow down to the physical implementation.

Method 4.1.2

Subdivision control in the PLCWT enables the accuracy-energy trade-off at the algorithmic level. Through top-down architectural and circuit optimizations, we implement the AQFE that translates these algorithmic trade-offs into hardware energy savings.

4.2 PLCWT Architecture

The AQFE implements the PLCWT method in hardware. This section reviews the algorithmic steps of the PLCWT and then presents the top-level module architecture showing the selected optimizations.

4.2.1 Algorithmic Mapping to Hardware

As detailed in Chapter 3, the convolution with the wavelet is simplified into a combination of signal integration and a weighted reduction (see Equation (3.11)). Algorithm 4.1 summarizes the stages of the PLCWT technique. The method consists of two main steps: (1) successive integration of the input signal using a trapezoidal method, and (2) a reduction step that multiplies subdivision coefficients with the integrated signal points and accumulates the results.

To produce N convolution points, the input signal X must be of length $3N$ to avoid boundary effects, assuming a convolution window of length $\leq 2N$. In practice, most ECoG systems have a large number of electrodes, thus the whole algorithm must be repeated on all input signals either sequentially or in parallel.

Algorithm 4.1: Feature Extraction using PLCWT

Input: Discrete ECoG signal $x[n]$ of length $3N$, J subdivision points $\{a_i\}_{i=1}^{J}$ of a wavelet support, border coefficients $\psi(a_1)$ and $\psi(a_J)$, precomputed coefficients $\{b_i\}_{i=1}^{J}$.

Output: Time-frequency components $\{S[\tau]\}_{\tau=0}^{N-1}$.

```
// Step 1: Double Integration of Signal Using Trapezoid Approach
```

$X^{(1)}[0] \leftarrow 0;$

$X^{(2)}[0] \leftarrow 0;$

for $n = 1$ **to** $3N - 1$ **do**

  $X^{(1)}[n] \leftarrow X^{(1)}[n-1] + (x[n-1] + x[n])/2;$

  $X^{(2)}[n] \leftarrow X^{(2)}[n-1] + (X^{(1)}[n-1] + X^{(1)}[n])/2;$

end

```
// Step 2: Dot Product using Subdivision Points
```

for $\tau = 0$ **to** $N - 1$ **do**

  `// 2a. Multiply by border values`

  $S \leftarrow \psi(a_J) \times X^{(1)}[a_J + \tau] - \psi(a_1) \times X^{(1)}[a_1 + \tau];$

  `// 2b. Reduce over the subdivision`

  for $i = 1$ **to** J **do**

    $S \leftarrow S + b_i \times X^{(2)}[a_i + \tau];$

  end

  $S[\tau] \leftarrow S;$

end

return $\{S[\tau]\}_{\tau=0}^{N-1}$;

The algorithm requires storage of the integrated signals $X^{(1)}$ and $X^{(2)}$, in addition to the precomputed wavelet coefficients b_i. As fixed-point multiplications require approximately 30× more energy than additions [60] (see 3.3.2) the energy cost is dominated by the number of multiplications in the reduction step, which scales with the subdivision size J. By adjusting the number of multiplications, we create a trade-off between the approximation quality and energy consumption.

4.2.2 Top-Level Block Diagram and Dataflow

Figure 4.3 shows the top-level block diagram of the AQFE, consisting of Random Access Memories (RAMs) and functional blocks. The input samples are provided by an upstream acquisition and pre-processing stage in the *Input RAM*, which decouples the feature extractor from the front-end. After processing, the extracted features are written to the *Output RAM*, from which they can be read by the downstream decoding model.

The architecture also includes dedicated coefficient memories. The *SpW0 RAM* stores the non-zero wavelet border values $\psi(a_1)$ and $\psi(a_J)$, while the *SpW1 RAM* holds the subdivision coefficients b_i. Both memories use a sparse format {index, value} to only store the non-zero coefficients, which reduces the storage requirements and avoids unnecessary memory accesses. Moreover, the *Integ RAM* stores the integrated signal $X^{(2)}$, which is reused during the reduction step.

In addition to the RAMs, the AQFE architecture comprises four functional blocks: three compute units and a control unit, implementing the different steps of the PLCWT

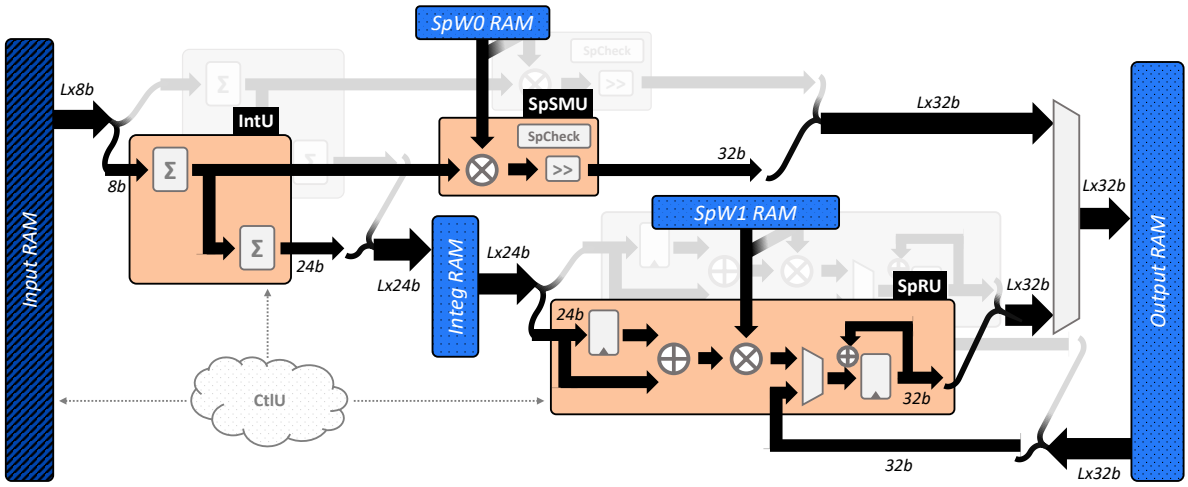


Figure 4.3: Top-Level Block Diagram of the AQFE Showing Compute Units (IntU, SpSMU, SpRU), the CtlU, and Associated Memories

algorithm.

- **Integrator Unit (IntU):** Computes the first and second antiderivatives $X^{(1)}$ and $X^{(2)}$ using trapezoidal integration (Step 1).
- **Sparse Shift Multiplication Unit (SpSMU):** Fetches border values from the *SpW0 RAM* and multiplies them with $X^{(1)}$ (Step 2a). Only a limited number of most significant bits are kept after the multiplication, corresponding to a right shift applied to the result.
- **Sparse Reduction Unit (SpRU):** Implements the subdivision-based dot product between the precomputed coefficients b_i and the samples of $X^{(2)}$ (Step 2b). For symmetric wavelets, values are summed or subtracted before multiplication, halving the number of required operations.
- **Control Unit (CtlU):** Coordinates the overall sequencing of the three compute units and manages memory accesses.

The AQFE can be implemented as a single-lane architecture in which data from all channels is processed sequentially, or as multi-lane configurations able to process L channels in parallel, increasing the throughput by a factor L, as shown as shown by the grey processing elements in Figure 4.3. In the latter, each lane contains its own IntU, SpSMU and SpRU, while sharing RAMs among the lanes. In Table 4.1, we show the different memory sizes needed for different configurations. For simplicity, we consider input signals with $3N$ samples of 8 bits (corresponding to N-sample buffers), recorded across C channels. Each input signal is convolved with F wavelets quantized over B bits. These input assumptions are based on the BCI system developped at CEA/Clinatec, which was used to collect data in [20]. The actual data occupies 8 bits, which corresponds to the selected input bitwidth of our module.

The *SpW1 RAM* holds $J/2$ coefficients for each wavelet, meaning that the maximum subdivision size is J, when exploiting wavelet symmetry. The *SpW RAMs* use a sparse format where each entry holds a coefficient (B bits), an index ($\log_2(3N)$ bits), and an additional control bit. When using L lanes, samples from L consecutive channels are stored in the same wordline for the *Input*, *Integ*, and *Output RAMs*. Note, the total size of the *Input RAM* remains constant regardless of the number of lanes, only the form factor changes. The number of channels C only affects the size of the *Input RAM*.

Table 4.1: RAM Sizes for Different AQFE Configurations

RAM	Width (bits)	Words
SpW0 RAM	$B + \log_2(3N) + 1$	$2 \times F$
SpW1 RAM	$B + \log_2(3N) + 1$	$(J/2) \times F$
Input RAM	$L \times 8$	$(3+1) \times N \times (C/L)$
Integ RAM	$L \times 24$	$3N$
Output RAM	$L \times 32$	$2 \times N \times F$

4.2.3 Architectural Optimizations

In addition to implementing the algorithmic optimizations of the PLCWT such as symmetry factorization and dynamic quality control, the architecture integrates further techniques to optimize area and energy consumption. These include memory re-use and multi-lane parallelism, described below.

4.2.3.1 Memory Re-use

Since the PLCWT algorithm uses the antiderivatives $X^{(1)}$ and $X^{(2)}$ to compute time-frequency components through convolution, these values need to be stored, and then multiplied with different wavelet coefficients. A simple approach would be to store both signals in two memories which would then be read during the reduction step. To reduce memory accesses, we propose an alternative solution, where the SpSMU multiplies the border coefficients read from the *SpW0 RAM* with the corresponding points from $X^{(1)}$, on the fly, avoiding an additional memory to store the first antiderivative $X^{(1)}$. The border product results are then temporarily stored in the *Output RAM*, to be later read by the SpRU when computing the reduction. After accumulating each convolution result, the SpRU writes this final value in the *Output RAM*, overwriting the border products written by the SpSMU. By re-using the memory in which these products were stored, it is possible to reduce the *Output RAM* size by 33%.

4.2.3.2 Multi-Lane Parallelism

As highlighted in the architecture, the AQFE can support different lane configurations to process data from L channels in parallel. For a configuration with L lanes, the size of the *SpW0* and *SpW1 RAMs* remains unchanged, but the number of read requests in both RAMs is reduced by a factor of L, as the coefficients are read once and shared across all

Table 4.2: RAM Read and Write Accesses per Inference in the AQFE Module

RAM	Read Accesses	Write Accesses
SpW0 RAM	$2 \times (2N) \times F \times (C/L)$	-
SpW1 RAM	$(J/2) \times N \times F \times (C/L)$	-
Input RAM	$(3N) \times (C/L)$	$N \times (C/L)^{(1)}$
Integ RAM	$J \times N \times F \times (C/L)$	$(3N) \times (C/L)$
Output RAM	$(2+1) \times N \times F \times (C/L)^{(2)}$	$(2+1) \times N \times F \times (C/L)$

(1) The write accesses are performed by the upstream acquisition and pre-processing stage.

(2) $N \times F \times (C/L)$ of the read accesses are performed by the downstream decoding model.

lanes. For other memories, wider word accesses amortize the per bit energy cost [107]. Furthermore, increasing L increases the feature throughput, reducing the clock frequency needed to meet the throughput of a given application, as will be shown next. More lanes, however, result in a larger area, as the *Integ* and *Output RAMs* are wider. In Table 4.2, we count the number of read and write operations performed in the AQFE during an inference window.

Conclusion 4.2.3

By combining memory reuse and multiple processing lanes in the multi-lane architectures, it is possible to reduce both the storage and number of RAM accesses. Also, reading larger words reduces the per bit energy cost.

4.2.4 Cycle Count and Operating Frequency

In the target system, the AQFE will be integrated into a platform with a fixed clock frequency, so the number of lanes must be configured to meet the throughput requirements at this frequency. In practice, increasing the number of lanes reduces the clock frequency required to preserve the overall performance. The number of cycles needed to process input signals depends on the selected quality (approximation parameter J) and the lane configuration. The worst case occurs at maximum quality, where the largest number of subdivision points must be processed. The cycle count also scales with the signal length and the decision rate. These parameters determine the minimum operating frequency of the AQFE module. For the design implementation, the maximum quality configuration is used to dimension the clock frequency to guarantee real-time operation. With L lanes, this frequency can be reduced by $1/L$, as multiple channels are processed in parallel. Alternatively, if the clock frequency is maintained constant, the throughput of the circuit increases by a factor L, when we increase the number of lanes.

4.3 Physical Implementation

This section details the AQFE physical implementation. We target a specific BCI application, first fixing the circuit parameters to meet its requirements and then characterizing the on-chip memory size and operation count for this application. Finally, we describe the circuit-level low-power optimizations, showing the resulting layouts for the single- and multi-lane configurations.

4.3.1 Circuit Parameters and Memory Characterization

We target a motor decoding application with 64 ECoG channels at 590 Hz and a feature (decision) rate of 10 Hz. For each data buffer, the AQFE computes the PLCWT per channel using 15 complex-valued wavelets, quantized over 17 bits (see 4.4.1). This results in 960 computed features per input data buffer. The choice of these parameters is inspired by the real-world BCI use-case in [20]. The approximation parameter J is dynamically selected with a maximum possible value of 32 subdivision points. We implement three variants of the architecture: a single-lane (**AQFE-1**), a two-lane (**AQFE-2**), and a four-lane (**AQFE-4**) implementation.

4.3.1.1 Memory Footprint Across Lanes

Figure 4.4 shows the distribution of on-chip memory for the five RAM blocks for the three AQFE configurations. In the single-lane implementation (**AQFE-1**), the total footprint is 30.8 KB, dominated by the *Input* and *Output RAMs*, which occupy 14.8 KB and 13.8 KB, respectively. The *SpW0*, *SpW1* and *Integ RAMs* account all together for 2.2 KB.

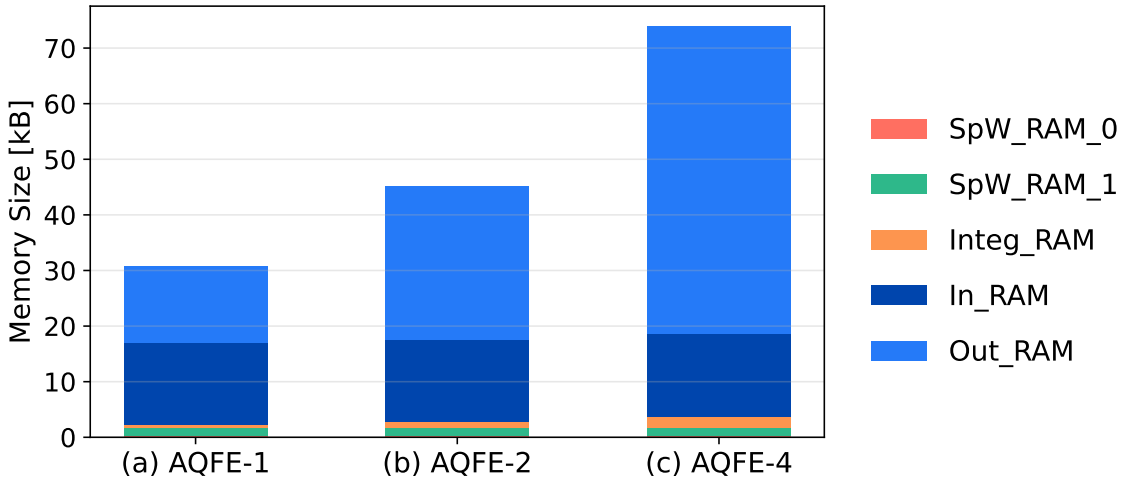


Figure 4.4: RAM Block Sizes for (a) **AQFE-1**, (b) **AQFE-2**, and (c) **AQFE-4**.

In the two-lane configuration (**AQFE-2**), the total rises to 45.2 KB, due to the doubled *Integ* and *Output RAM* widths. For the four-lane version (**AQFE-4**), the total reaches 73.9 KB, with an *Output RAM* size of 55.3 KB, and an *Integ RAM* of 2.1 KB. The *SpW* memories are fixed across all configurations, as they store the wavelet coefficients. Similarly, the *Input RAM* size remains constant with the number of lanes.

4.3.1.2 Memory Accesses

Figure 4.5 shows the total number of bytes read or written per inference window, combining all accesses across the RAM blocks. This metric represents the data movement within the AQFE when processing a single input buffer.

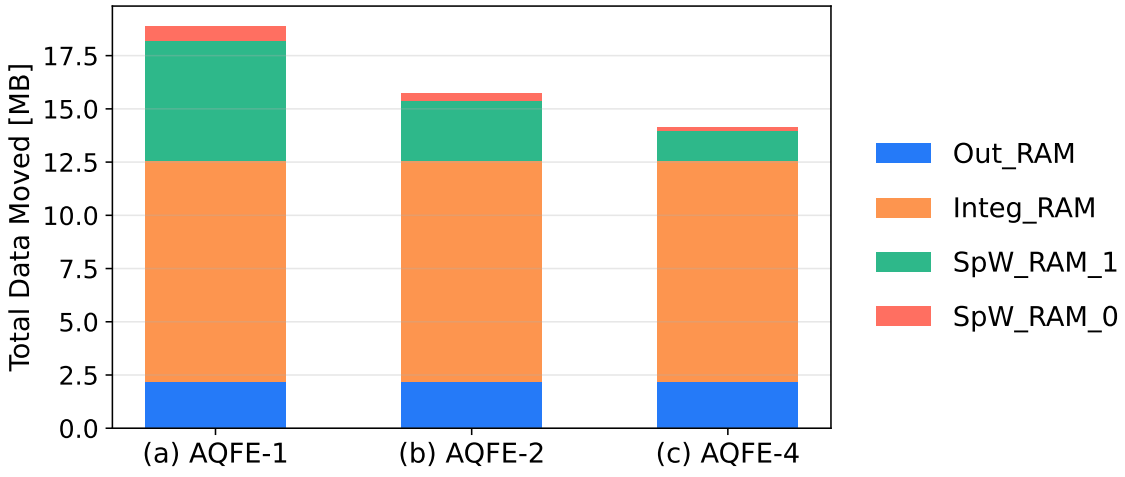


Figure 4.5: Bytes Accessed per RAM for (a) **AQFE-1**, (b) **AQFE-2**, and (c) **AQFE-4**.

Each access corresponds to a full-width RAM operation, so the total byte volume is obtained by multiplying the number of accesses by the memory width. As reported in Table 4.2, the number of operations in memories is dominated by read accesses. In this analysis, the *Output RAM* read requests performed by the downstream decoding model are excluded. The *Input RAM* operations are also omitted as they only account for a negligible portion of the read data (11.3 KB) per window.

Across all configurations, the *Integ RAM* dominates the traffic with approximately 10.9 MB of data read per window. The *Output RAM* data accesses remain constant at 2.27 MB, while the *SpW0* (resp. *SpW1*) *RAMs* read data is reduced from 736 KB (resp. 5.89 MB) to 184 KB (resp. 1.47 MB) as the number of lanes increases from one to four. Overall, the data volume per inference window decrease from 19.8 MB in **AQFE-1** to 14.8 MB in **AQFE-4**. These results, combined with the memory sizes reported in Figure 4.4, provide a detailed view on how the lane configuration affects both the memory footprint and data accesses.

Conclusion 4.3.1

As the number of lanes L increases, the total storage (area) increases, but the number of memory accesses decreases. The analysis highlights a trade-off between memory size and accesses, which will be later linked to leakage and dynamic power contributions.

4.3.2 Circuit-Level Optimizations

While the algorithm reduces arithmetic complexity and the architecture balances memory usage and parallelism, the physical implementation further minimizes power at the circuit level. As shown in Section 4.3.1, the data access evaluation was used to dimension the memory cuts and select the most energy-efficient macro configurations for each *RAM*. Memories with the highest access rates, such as the *Integ* and *SpW RAMs*, were implemented using cuts that minimize the per-access energy. Smaller and less frequently accessed memories such as the *Input* and *Output RAMs* use cuts that minimize leakage.

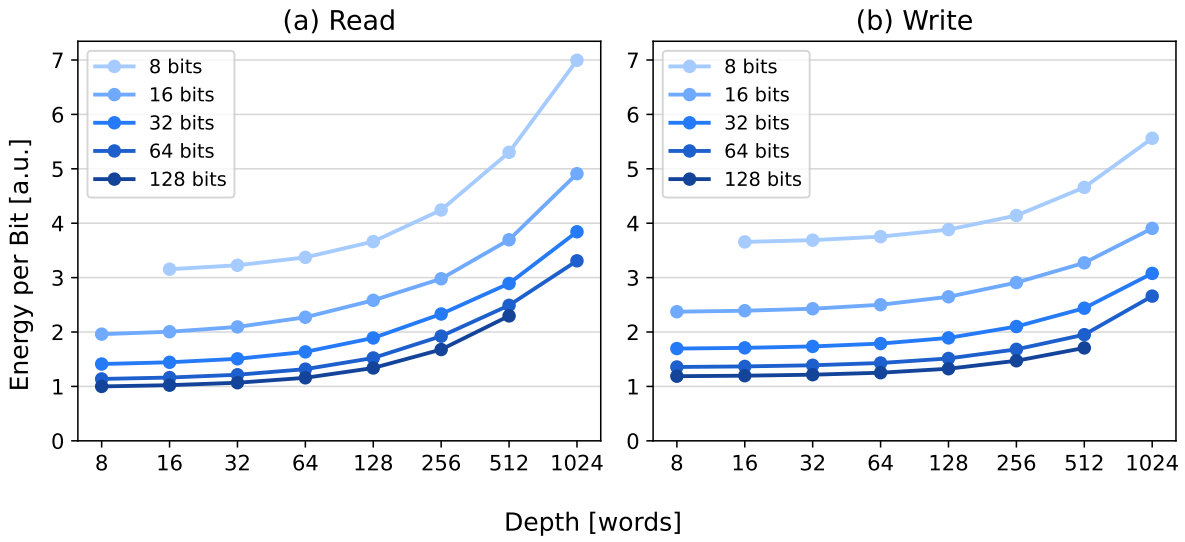


Figure 4.6: Normalized Energy per Bit Access for (a) Read and (b) Write Accesses as a Function of Depth for Small RAM Macros, Using Different Memory Cut Widths.

As shown in Figure 4.6, for RAM macros with depths from 8 to 1024 words, the per-bit access energy increases with depth for both read and write operations. This is due to longer physical bitlines, which increases the effective capacitance. The higher capacitance results in a higher switching energy per access. At a fixed depth, wider word widths consistently lower the access energy per bit, as peripheral overheads are amortized across more bit lines. All energy and power values are reported in normalized arbitrary units (a.u.) to protect intellectual property. Large RAM macros (4k to 32k words), shown in Figure 4.7, have similar per-bit energy profiles.

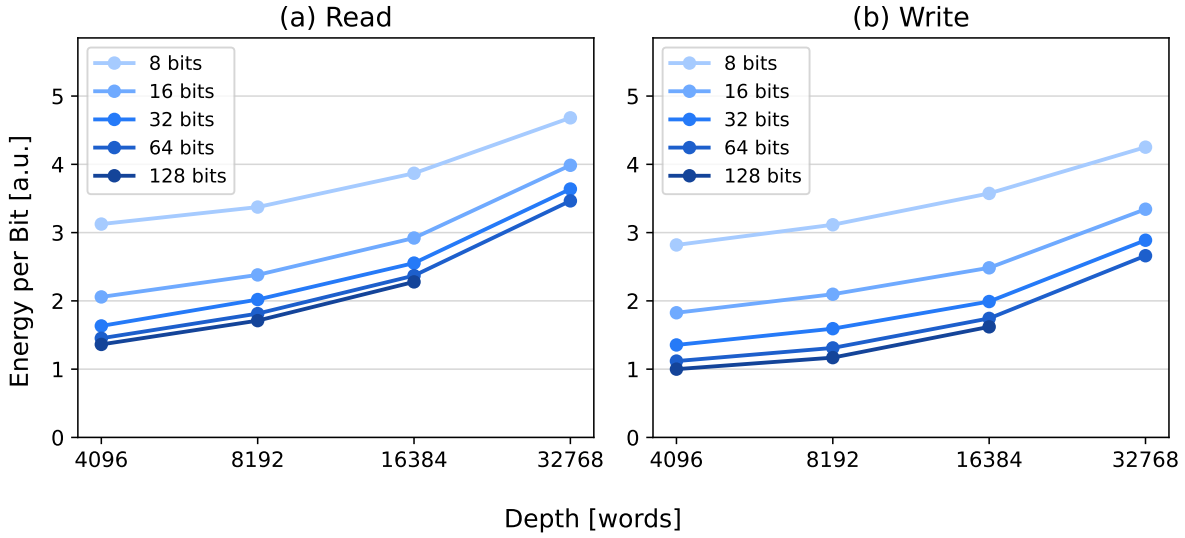


Figure 4.7: Normalized Energy per Bit Access for (a) Read and (b) Write Accesses as a Function of Depth for Large RAM Macros, Using Different Memory Cut Widths.

The memory macros are generated by different generators in the same technology, their names are not disclosed in this manuscript. In Figure 4.8, we observe that the RAM leakage power per bit decreases with depth, and with wider words at a fixed depth, since peripheral leakage is amortized across more bits.

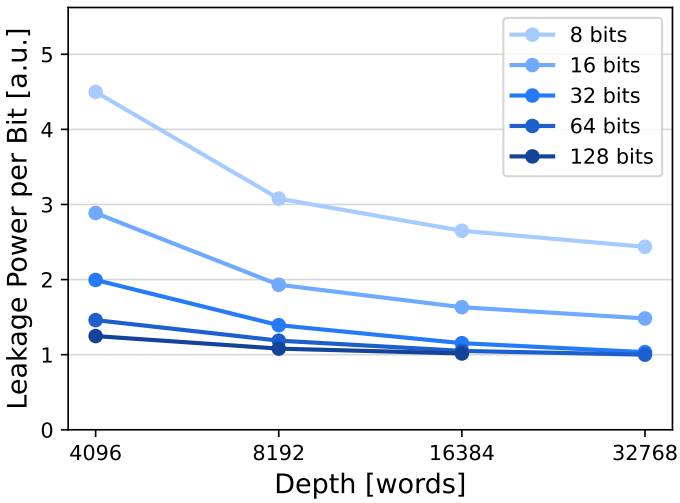


Figure 4.8: Leakage Power per Bit as a Function of Depth for Large RAM Macros, Using Different Memory Cut Widths.

In addition to RAM macro optimization, the micro-architecture was carefully pipelined so that timing closure could be achieved using only ultra-high threshold voltage (UHVT28) standard cell libraries [29], to reduce leakage power. Careful RTL coding enabled the synthesis tool to automatically infer clock gates on approximately 96% of the registers. In addition, unused RAM macros were manually clock-gated, preventing unnecessary toggling and further reducing dynamic power.

Furthermore, we approximate the arithmetic operations, mainly multiplications, using a fixed-point stage-specific precision. Configurable shifting and bit truncation allow tuning to minimize the number of significant bits required to maintain high accuracy. By using custom widths, the synthesis tool simplifies the datapath by propagating constants and pruning logic that feeds the dropped slices.

Finally, as detailed in 4.2.4 the operating frequency is chosen according to the lane configuration. For the highest approximation quality (32 subdivision points), real-time performance is achieved at approximately 50 MHz for the single-lane configuration, while **AQFE-2** and **AQFE-4** achieve the same throughput at 25 MHz and 12.5 MHz, respectively. This scaling directly reduces dynamic power, as it decreases clock activity across the datapath.

Conclusion 4.3.2

During the physical design phase, the selection of UHVT-only standard cells and energy-efficient RAM macros, combined with a RTL style that inferred clock-gating for 96% of flip-flops, enabled a power-optimized circuit.

4.3.3 Implementation Flow

The three AQFE variants were implemented using a standard industrial design flow in a 22 nm FDSOI technology [29]. Each configuration (**AQFE-1**, **AQFE-2**, and **AQFE-4**) follows the same implementation methodology, differing only in the number of parallel lanes and the corresponding memory cuts.

Logic synthesis was performed using Synopsys Design Compiler™ with timing and power optimizations. Post-synthesis netlists were then imported into Cadence Innovus™ for placement, routing, and clock-tree synthesis. All designs were implemented with a supply voltage of 0.8 V and closed timing using only UHVT28 standard cells to minimize leakage power. Memory instances were instantiated from foundry-provided Static Random Access Memory (SRAM) generators.

Functional validation was performed through RTL simulations and post-layout gate-level simulations with back-annotated timing (SDF) and parasitics (SPEF), using ECoG input signals from actual datasets provided by Clinatec. The resulting activity (VCD) files were used for time-based power analysis in Synopsys PrimePower™. Accordingly, the power data are derived from real toggle activity with full timing and parasitic back-annotation, rather than heuristic estimation. This flow ensures comparable results to evaluate trade-offs between area, power, and performance for the different configurations.

4.3.4 Layout of the Three Implementations

All variants were implemented in 22 nm FDSOI using the flow described in Section 4.3.3. Figure 4.9 shows the layout of the **AQFE-1** implementation. This configuration (**AQFE-1**) occupies 0.061 mm^2 and is dominated by the *Input* and *Output RAMs*. Memory

dominates the total area, and the compute and control logic are placed between the RAM macros. The layout reflects the partitioning described in Section 4.3.1.1.

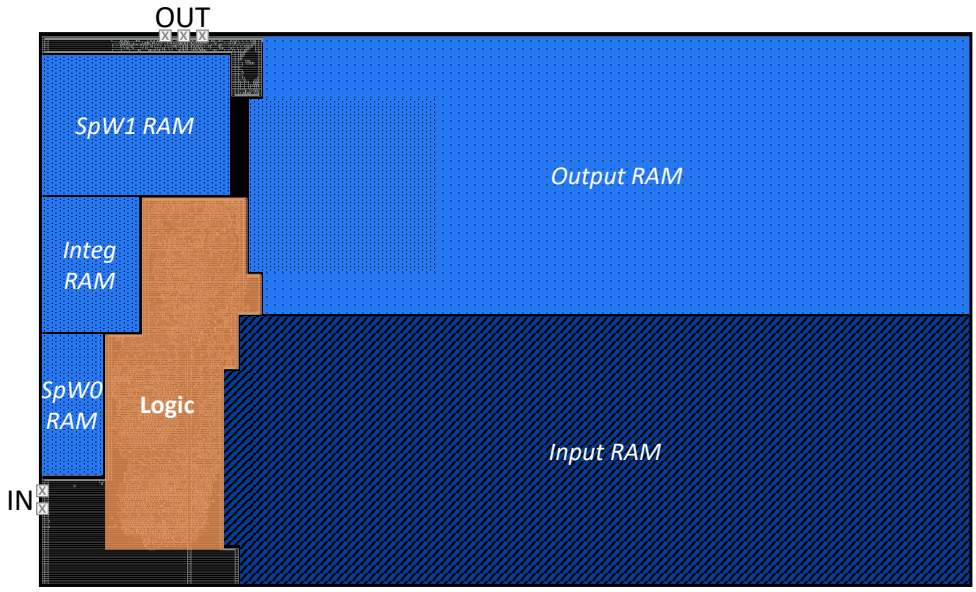


Figure 4.9: Post-Layout View of **AQFE-1** in 22 nm FDSOI.

The two multi-lane implementations are shown in Figures 4.10 and 4.11. These variants occupy 0.088 mm^2 for **AQFE-2** and 0.132 mm^2 for **AQFE-4**, with the increase mainly driven by the wider RAMs. A quantitative comparison of the three implementations is presented in Section 4.4.

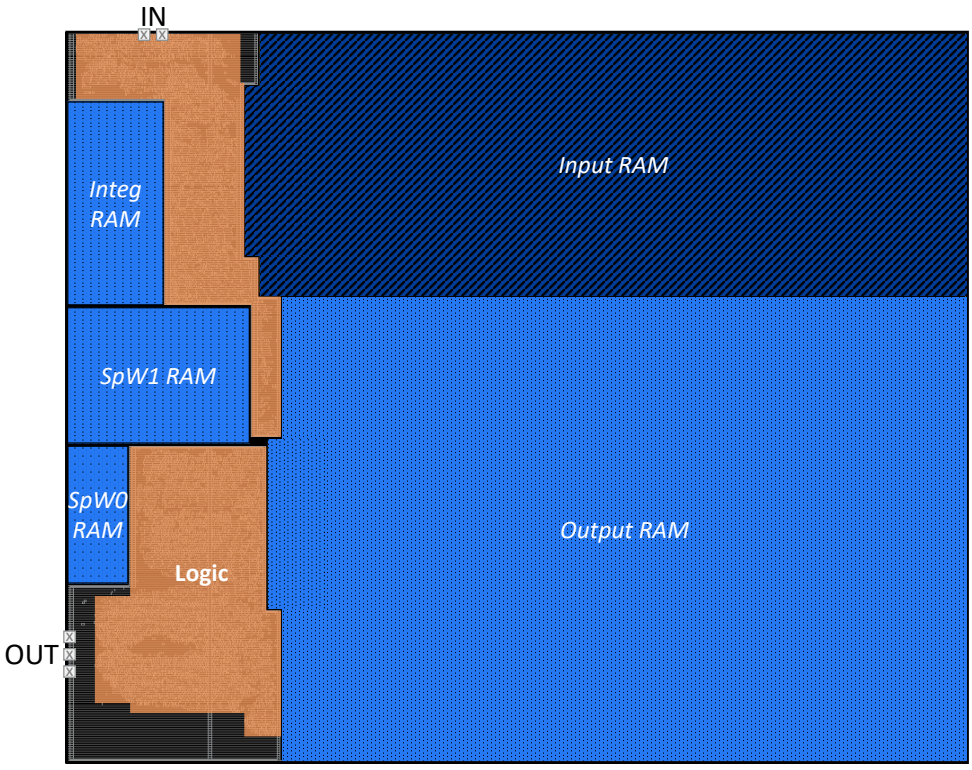


Figure 4.10: Post-Layout View of **AQFE-2** in 22 nm FDSOI.

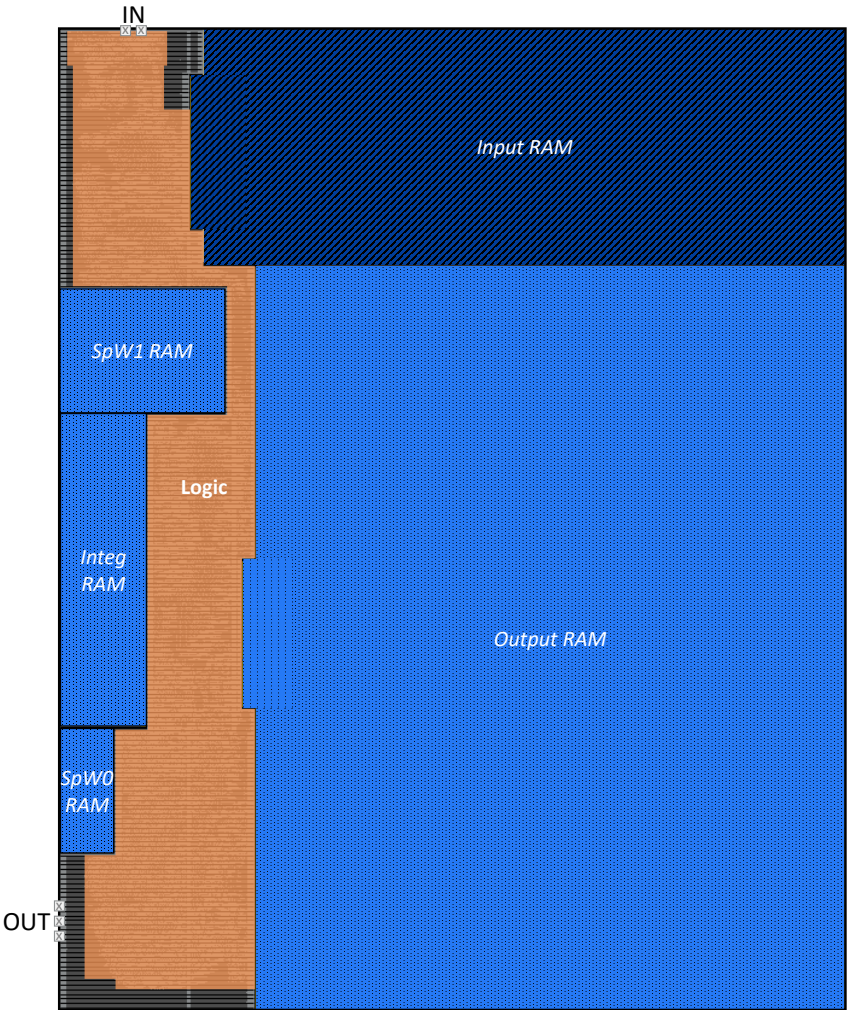


Figure 4.11: Post-Layout View of **AQFE-4** in 22 nm FDSOI.

4.3.5 Approximate Multipliers

Approximate computing trades precision for energy, and induced errors are analyzed, propagated through the signal chain, and evaluated at the task level [24]. As the compute bottleneck in our circuit is in the multiplications, we explored approximate multipliers [66], more specifically Broken-Array Multipliers (BAMs) [109], to further reduce power. We first ran algorithmic experiments to sweep the horizontal h (truncation of the h least-significant rows) and vertical v (truncation of the v least-significant partial-product columns) truncation parameters. For each (h, v) setting we re-computed the features and measured the impact on accuracy, and then, we selected a (h, v) configuration that we implemented in a full RTL flow with synthesis and power estimation. The outcome, however, was not conclusive due to the strong optimization of baseline industrial multipliers. We provide preliminary results and methodology details in Appendix D.

4.4 Results and Discussion

This section presents the evaluation results of the AQFE implementations. In the following, three quality parameters (Low **LQ**, Medium **MQ** and High **HQ**), corresponding to J values of 4, 18 and 32, are tested. We first describe the reference model used for

functional validation with the algorithmic baseline presented in Section 3.5. Furthermore, we report the power consumption from post-layout simulations and analyze the trade-offs between approximation quality and energy efficiency. Finally, we compare the proposed design to existing BCI feature extractors to highlight its advantages.

4.4.1 Reference Model and Functional Validation

A reference model was implemented in C to accurately reproduce the behavior of the AQFE circuit. This model performs the same operations as in the hardware datapath. All arithmetic operations are executed in fixed-point format, matching the quantization and shifting implemented in the RTL.

The goal of this model is to functionally validate the hardware behavior prior to synthesis, and to evaluate the degradation of accuracy induced by the quantization in the hardware model, compared to the algorithmic model in Section 3.5. No change is introduced in the mathematical formulation of the PLCWT itself. The only variable at this stage is the custom fixed-point quantization applied to the arithmetic operations. The analysis was performed using ECoG data from the virtual avatar dataset presented in Section 3.4.1.2.

To select the best quantization of the wavelet coefficients, we report the Correlation Coefficient (CC) between the features comptuted by the 32-bit fixed-point PLCWT reference and those extracted from the behavioral model of the circuit. Figure 4.12 shows the mean CC at **HQ** as a function of the wavelet coefficient bitwidth. The correlation reaches 0.96 from 17 bits upward, giving a threshold value for wavelet quantization.

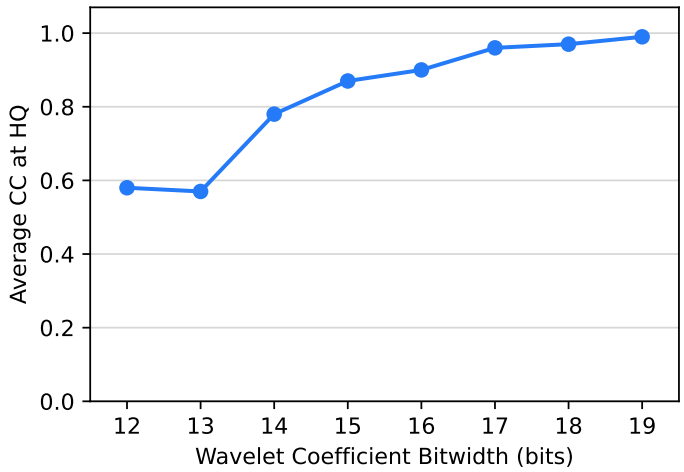


Figure 4.12: Average Correlation at HQ as a Function of the Wavelet Coefficient Bitwidth.

To better quantify the effect of quantization, we compute the CC between the complex outputs generated by the 32-bit reference model and those obtained from the circuit. We also compute the correlation between the complex module of these complex values, which corresponds the extracted features. The CC is computed across all extracted frequencies and approximation quality levels, and the results are shown in Figure 4.13.

A slight degradation appears at higher frequencies, as the extracted components have lower magnitude compared to low-frequency bands. This can be explained by the signal power decrease with $1/f$ [48]. In practice, the reduction step yields smaller results that are more sensitive to quantization noise in the least significant bits. This analysis of

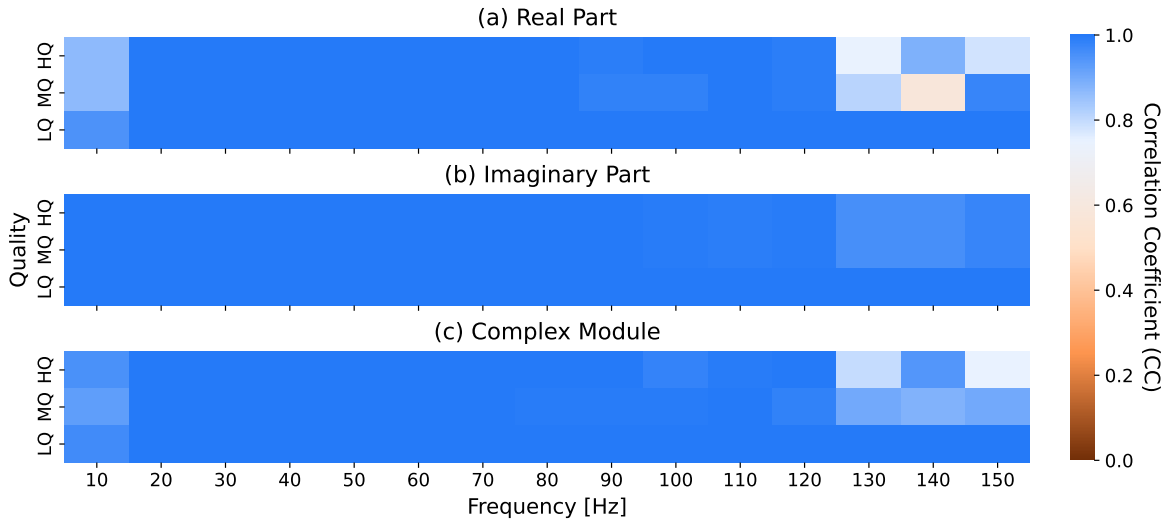


Figure 4.13: Correlation of the (a) Real, (b) Imaginary Part, and (c) Module of the PLCWT Complex Components across Different Frequencies and Approximation Levels.

correlation allows us to validate the selected quantization parameters for both the wavelet coefficients (17 bits) and the arithmetic operations, to preserve feature integrity. The mean CC across all frequencies, between the features extracted by the AQFE and those computed in software, is 0.99 at **LQ**, 0.97 at **MQ** and 0.96 at **HQ**. For the selected parameters, the correlation remains thus close to 1 for the extracted frequencies across all tested qualities. These results confirm that the proposed quantization reduces hardware costs, while achieving similar performance to the 32-bit reference algorithmic model.

Conclusion 4.4.1

A CC between above 0.96 confirms that the custom width fixed-point implementation closely matches the PLCWT algorithmic reference.

4.4.2 Power Analysis

Power was estimated in Synopsys PrimePower™ from gate-level simulations with back-annotated timing, as described in Section 4.3.3. The selected parameters are those detailed in Section 4.3.1. Figure 4.14 reports the time profiles of total and leakage power for the three implementations when processing 100 ms data buffers.

The three quality paramters (**LQ**, **MQ** and **HQ**) are tested on consecutive buffers. The total power increases during the active portions of each window due to dynamic switching. The clock frequency for each implementation is selected to guarantee real-time operation at the worst case **HQ**, as detailed in 4.2.4. The frequency selected for the single-lane implementation is two (resp. four) times higher than that of the two-lane (resp. four-lane) implementation, which can be seen in the power profiles. In simpler words, for a computation window, the power profile of the **AQFE-1** is similar to that of two concatenated and time-shrinked (by 2×) computation windows of the **AQFE-2**.

For all lane configurations, the lower-quality buffers (**LQ** and **MQ**) have shorter compute time as fewer subdivision points are used. The compute time goes down from 88 ms at **HQ** to 25 ms at **LQ**, achieving a 3.5× reduction. In these configurations, during the idle period, the power drops down to a value that is only slightly above the leakage power,

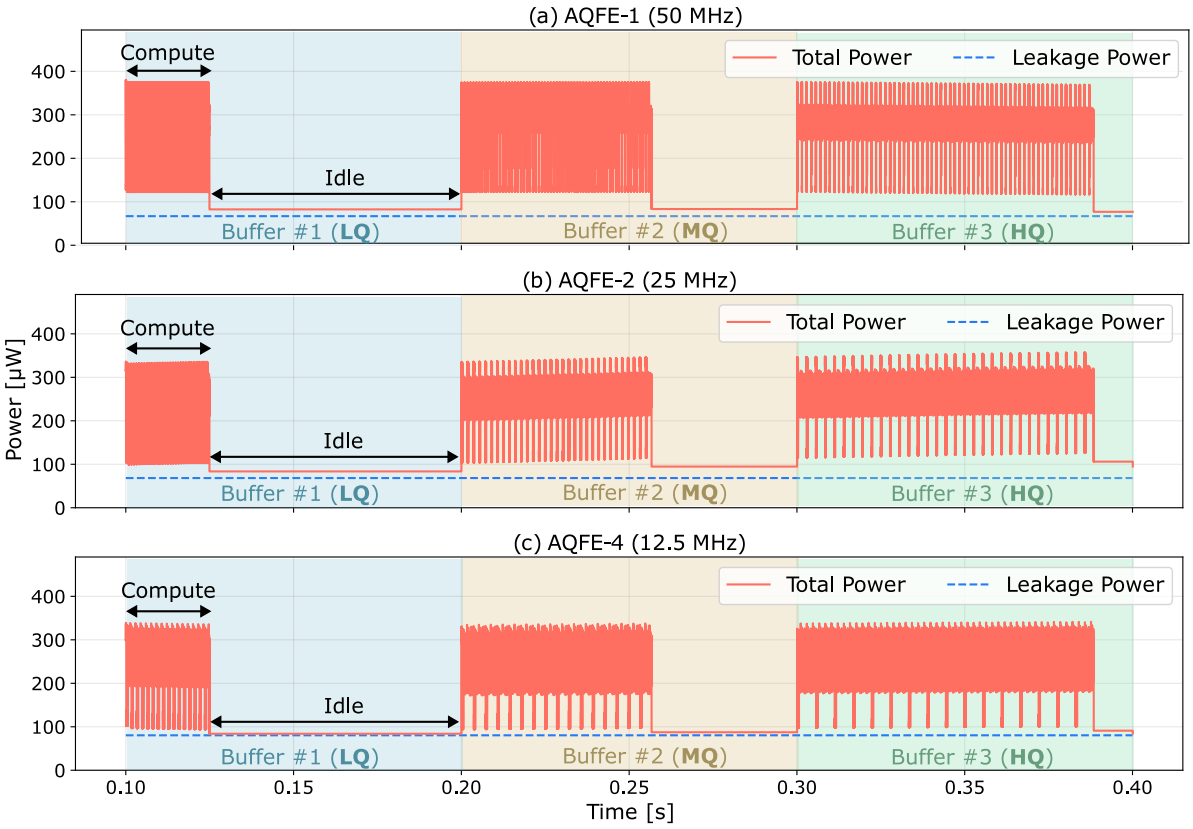


Figure 4.14: Leakage and Total Power Profiles for the (a) **AQFE-1**, (b) **AQFE-2**, and (c) **AQFE-4** Implementations.

due to the effective clock gating. A zoomed view of the **AQFE-4** profile, in Figure 4.15, shows the power variations within a processing window.

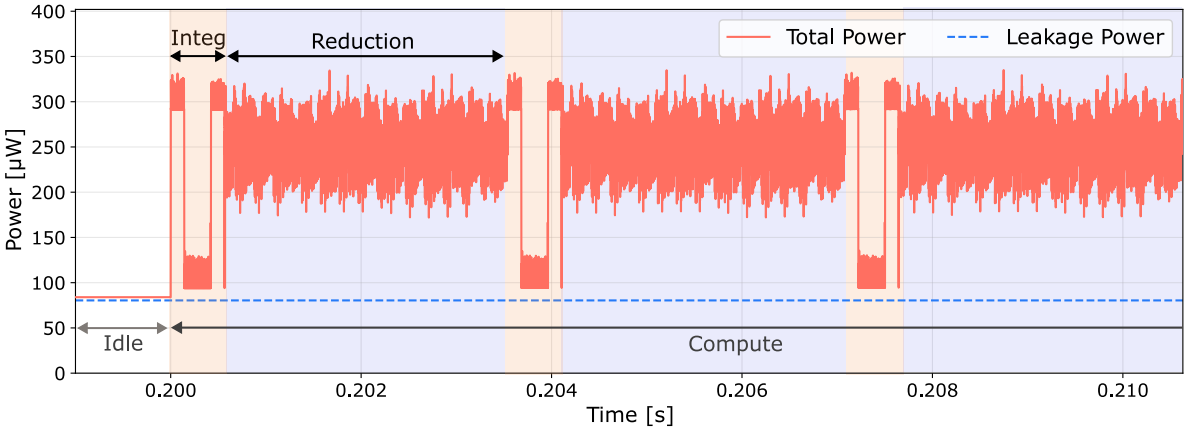


Figure 4.15: Zoomed Leakage and Power Profile for the **AQFE-4** at **MQ**.

During the integration step, the IntU performs trapezoidal accumulations. Two power consumption peaks appear at the beginning and the end of this step when the SpSMU multiplies the border coefficients. Between these peaks only additions are performed, hence the lower consumption. During the reduction step, the SpRU computes dot products, resulting in a higher power consumption.

Conclusion 4.4.2

Across all lane configurations, algorithmic quality control reduces the compute time by up to 3.5×, which translates directly into circuit-level energy savings.

4.4.3 Power-Decoding Performance Trade-offs

For each subdivision size J and lane configuration, we run gate-level simulations for which we obtain power profiles as shown in 4.4.2. We then compute the average power as follows:

$$P(J) = \frac{E_{\text{buf}}(J)}{T_{\text{buf}}}, \tag{4.1}$$

where $E_{\text{buf}}(J)$ is the energy consumption required to process an input buffer, obtained by integrating the power curve over T_{buf} (100 ms), the buffer duration. In the following, we refer to this as the overall power consumption.

4.4.3.1 Power Consumption vs Subdivision Size (J)

Figure 4.16 shows how the overall power scales with the subdivision size J. Similar profiles are observed across all lane configurations: power increases linearly with J, proportional to the number of multiplications in the reduction step. A linear fit of the total power versus J yields slopes of 4.9 μW/J, 3.6 μW/J, and 4.3 μW/J for **AQFE-1**, **AQFE-2**, and **AQFE-4**, respectively.

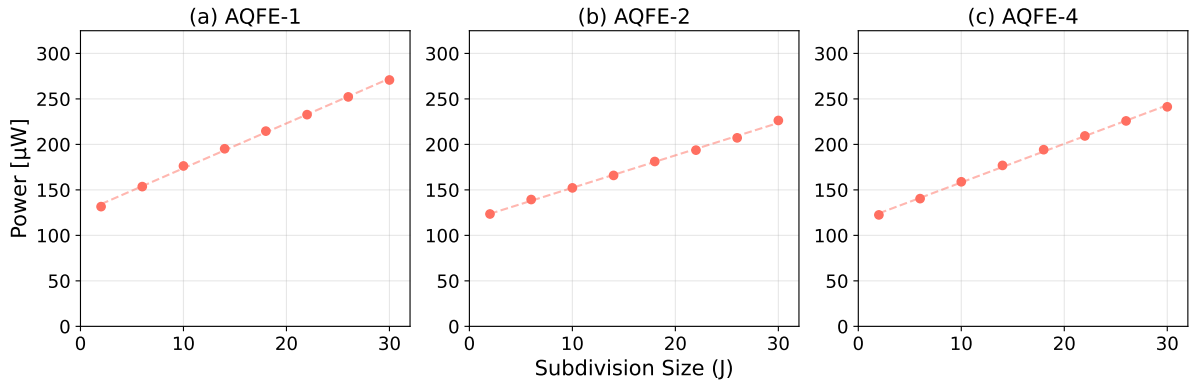


Figure 4.16: Average Power vs. Subdivision Size J for (a) **AQFE-1**, (b) **AQFE-2**, and (c) **AQFE-4**.

These plots confirm that the proposed circuit indeed consumes less power, when the quality parameter is reduced. The slopes vary with the number of lanes, reflecting different leakage and dynamic power consumption across the three configurations.

4.4.3.2 Power breakdown for Physical Implementations

Despite the similar overall power profiles, the number of lanes determines other circuit parameters. To better understand the influence of this variable, we decompose total power into leakage (static) and dynamic components. Figure 4.17 shows the power breakdown of the AQFE module for the three lane configurations at different qualities.

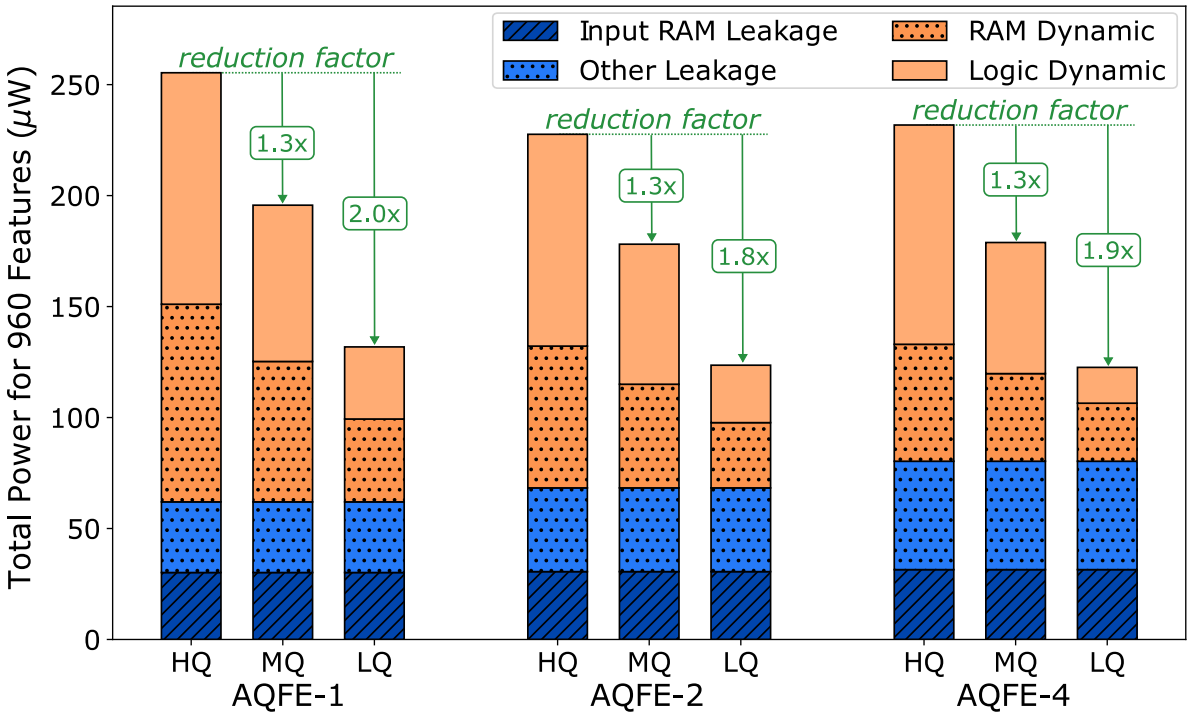


Figure 4.17: Power Composition Across Lane Configurations for Different Quality Parameters.

When reducing the quality from **HQ** to **LQ**, the overall power decreases for all configurations. For the **AQFE-1** configuration at **HQ**, 75% of the power consumption is dynamic, including logic (arithmetic operations and control) and memory accesses. When reducing the quality from **HQ** to **LQ**, the total power drops from 255 μW to 128 μW, achieving a 2× reduction.

Multi-lane versions benefit from lower operating frequency, L times fewer RAM accesses, and a lower per-bit energy cost for wider RAMs, reducing the RAM dynamic power. At **HQ**, **AQFE-2** (resp. **AQFE-4**) consumes 18% (resp. 22%) less dynamic power than **AQFE-1**. As a result, the multi-lane configurations achieve comparable power to the single lane despite having larger *Integ* and *Output* RAMs, thus higher leakage. More precisely, at **HQ**, multi-lane circuits consume slightly less power than the single lane, and they are able to reach the same low power at **LQ**.

These findings highlight the trade-off between different levels of parallelism. Multi-lane designs benefit from reduced operating frequency, while the single lane occupies less area. The total power remains comparable across all configurations, especially when operating at **LQ**. The power reduction achieved through quality control remains effective even when processing up to four lanes in parallel.

Conclusion 4.4.3

The average power scales linearly with the subdivision size J for all implementations. In multi-lane configurations, although leakage increases with larger memories, the dynamic power drops with lower clocks and shared RAM accesses, yielding a comparable overall power.

4.4.4 Comparison to Existing BCI Accelerators

4.4.4.1 SoA Feature Extractors

We compare our circuit with existing State-of-the-Art (SoA) feature extractors based on Fast Fourier Transform (FFT), filters, CWT, and other custom methods. As shown in Table 4.3, the power consumption per feature, at high precision, is at least 4× lower than that of existing digital circuits. The module is able to extract up to 960 features, 8× more than prior extractors, at a rate of 10 Hz, providing the required throughput for motor decoding [20]. The best implementation (**AQFE-1**) for this BCI use case has a minimal silicon area of 0.061 mm^2, 2× smaller than existing circuits. A large part of this area reduction can be attributed to the more advanced technology node.

Table 4.3: Summary of SoA Brain Signal Decoding Hardware Implementations

Ref. (Year)	Ch.	Features			Power (μW/feature)	Area (mm^2)	Scaled[(1)] Area (μm^2/feature)	Technology
		Method	Nb.	Rate				
2010 [32]	16	Custom	16	**10 Hz**	1.4	2.10	10096	ASIC (90 nm)
2011 [156]	4	FFT	1	2 Hz	0.99	8.88	317142	ASIC (130 nm)
2014 [180]	7	DCT	42	3.3 Hz	17.1	-	-	FPGA
2016 [5]	62	PCA	3	2 Hz	50666	-	-	FPGA
2016 [175]	4	FFT	1	0.5 Hz	34000	100	-	DSP
2017 [30]	1	CWT	1	1 Hz	47300	1.12	-	FPGA
2019 [90]	10	Filter	3	1 Hz	8333	-	-	FPGA
2019 [174]	7	Filter	1	1.3 Hz	150000	49	-	MCU
2020 [93]	32	Analog	32	-	0.205	0.145	54.6	ASIC (180 nm)
2022 [149]	64	Custom	64	4 Hz	1.42	0.12	284	ASIC (65 nm)
2024 [144]	**192**	Filter	128	1 Hz	1.68	0.71	840	ASIC (65 nm)
2025 [10]	32	PAC/PLV	48	-	0.88-1.92	5.46	17234	ASIC (65 nm)
AQFE-1	64	PLCWT	**960**	**10 Hz**	0.13-0.27	**0.061**	63.5	ASIC (22 nm)
AQFE-2	64	PLCWT	**960**	**10 Hz**	**0.13-0.24**	0.088	91.7	ASIC (22 nm)
AQFE-4	64	PLCWT	**960**	**10 Hz**	**0.13-0.24**	0.12	125	ASIC (22 nm)

[(1)] ASIC area is scaled to the 22 nm technology node using the factors in [157]. All values are normalized per feature.

Our design achieves a balance between area, energy efficiency, and adaptability. While previous works use static configurations, our architecture supports real-time approximation adjustment to meet dynamic quality requirements. This tuning mechanism reduces the power consumption by an additional factor of 1.8× to 2×, and offers similar power savings even when increasing the number of parallel lanes. For our BCI application, multi-lane implementations reach the same power per feature as the single lane, while reducing the system clock frequency. These implementations can also be advantageous when processing a larger number of input channels while maintaining a constant operating frequency.

Conclusion 4.4.4

Compared to prior digital feature extractors, the **AQFE** achieves 4× lower power per feature at high precision while extracting 960 features at 10 Hz. Dynamic quality tuning provides an additional 2× power reduction, for all lane configurations.

Chapter 5

AQFE-Based Full BCI System: Challenges and Perspectives

Building on Chapter 4, which detailed the Adaptive Quality Feature Extractor (AQFE) hardware and its energy-accuracy trade-offs, this chapter explores how such a module could be integrated into a Brain-Computer Interface (BCI) system. We propose a full decoding system including a closed-loop feedback that adjusts the quality of feature extraction at runtime. The mechanism is driven by Feature Evaluation Parameters (FEPs), applied either globally or selectively on a per-feature basis. These parameters are either internal indicators (periodic exact-feature checks or decoder confidence) or external sources of feedback (user satisfaction or secondary sensor inputs). This chapter is organized as follows:

- Section 5.1 describes the full decoding pipeline, the placement of the AQFE between the acquisition and decoding steps, and the feedback loop description.
- Section 5.2 gives closed-loop control examples driven by internal and external FEPs.
- Section 5.3 outlines the remaining challenges for deploying such a closed-loop BCI system.

Parts of this chapter are described in a French patent application (FR2501705).

5.1 Full-BCI Decoding System

We target a real-time BCI in which neural signals are transformed into motor commands. The system follows the typical three-stage pipeline, previously shown in Figure 2.1, with an additional runtime control module (controller) that closes the loop, as shown in Figure 5.1

The acquisition front-end streams neural samples to the AQFE, which computes complex convolutions at a selected quality level Q. If energy features are used, the Complex-to-Real (C2R) post-processing block, detailed in Appendix C, computes per-band energy with a selectable norm over custom windows, and forwards these features to the decoder. Finally, the decoder uses these features to produce motor commands.

At the end of each feedback window, the controller uses one or more internal or external FEPs to compute an evaluation metric. The obtained value serves as an input for a Lookup Table (LUT) that selects the AQFE quality Q for the next window. It is essential to define both the quality parameter Q and the FEPs that control it through feedback. For the Piecewise Linear Continuous Wavelet Transform (PLCWT), the quality

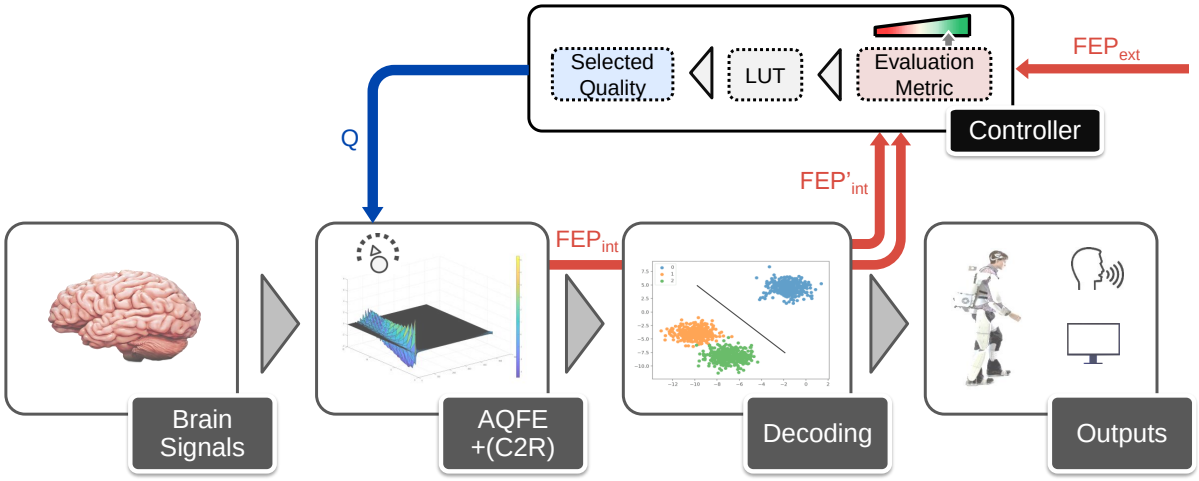


Figure 5.1: Closed-Loop BCI Decoding with Runtime Quality Control Using the AQFE.

selection corresponds to choosing a subdivision size J. Examples of possible FEPs will be detailed in Section 5.2.

This approach can be viewed in the same feedback-control framework as the BCI system described in [179], where decoder parameters are tuned based on closed-loop task performance rather than offline tuning. In our case, the design parameter is the feature quality: instead of setting Q manually, it is adjusted according to closed-loop performance, so that the system trades off accuracy and energy using the user-decoder interaction.

5.2 Adaptive-Quality Control Loop

The adaptive-quality control loop adjusts the extraction quality of the AQFE at each feedback window to balance decoding performance and energy. For the PLCWT, the quality is linked to the subdivision size J, but the concept remains applicable to other extractors with a tunable parameter. This parameter can be tuned using the FEPs, either globally for all features, or per specific electrodes/bands. The FEP can be evaluated as an internal (periodic exact-feature checks, decoder's confidence level) or external signals (user satisfaction, secondary sensors).

5.2.1 Global vs Per-Feature Control

The choice between global and per-feature control depends on the nature of the FEP, which modulates the quality Q. Typically, when the FEP is global, the adjustment is applied to all features at once. However, when the FEP is local, each feature's quality is tuned individually (per electrode and per band). The two tuning mechanisms are shown in Figure 5.2. They are illustrative rather than exhaustive and represent proposed use cases.

In both configurations, the global system consists of C recording channels, an AQFE with a bank of Feature Extractors (FEs) between *Input RAM* and *Output RAM*, an optional C2R block, a decoder, and one or more controllers. Channels write samples to the *Input RAM*, which is then read by the AQFE and processed by the FEs to compute the PLCWT. In the figure, each feature is represented as the output of a FE, and we show a bank of N FEs. In practice, the same physical FE (or a small set) can be time-multiplexed

across features. The complex outputs of the PLCWT are written in the *Output RAM*. The optional C2R module then computes energy features for a chosen norm and time window. Features are written to the *Output RAM* and read by the downstream decoder.

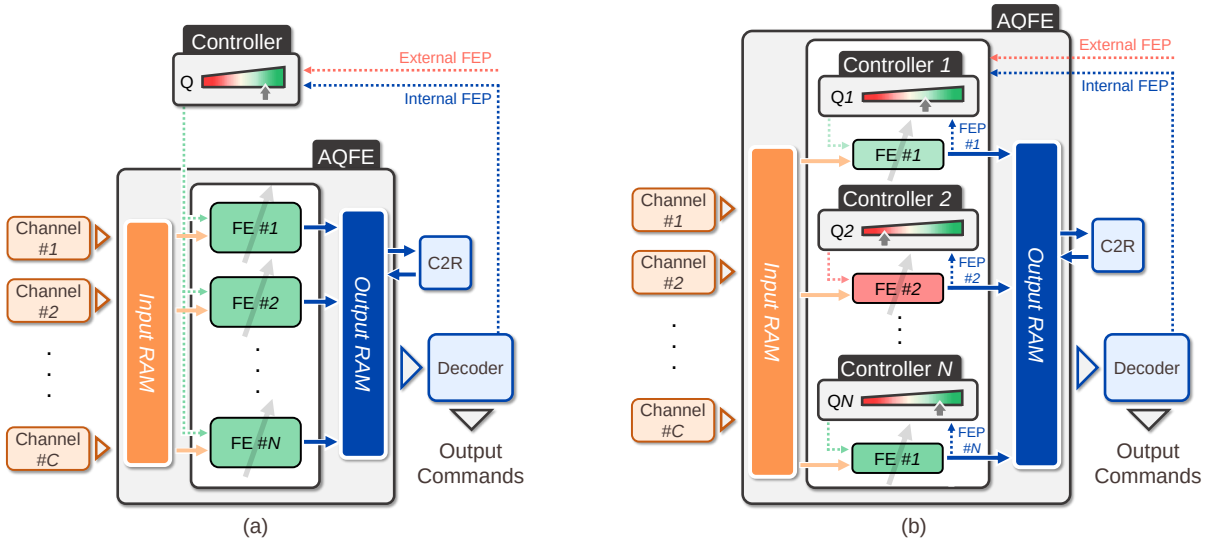


Figure 5.2: Quality Adjustment Using (a) Global and (b) Per-Feature Control.

Global control, shown in Figure 5.2 (a), applies one quality Q to the full feature set at once. This quality can be evaluated using one or more internal or external FEPs, using a controller that selects a discrete level Q and sends it to the AQFE. For the PLCWT, the global Q is then translated into per-feature subdivision parameters J_k via a programmable lookup table, meaning that the same Q does not necessarily imply the same J_k for all features. This allows each feature to have its own mapping between Q and J_k, to match frequency-dependent difficulty, detailed in Section 3.5.1.1.

Per-feature control, shown in Figure 5.2 (b), assigns a quality Q_k to each feature (electrode and band) per closed-loop feedback. A controller reads local FEP_k, evaluates a defined metric, and then updates Q_k using a LUT. In the diagram multiple controllers are shown for clarity, but in hardware a single controller (or a small set) can be time-multiplexed to select per-feature qualities. The mixed-quality outputs are fed through the C2R to compute features that are used by the decoding model.

5.2.2 Internal Quality Evaluation

We introduce internal FEPs used to modulate the quality during feedback. We consider two approaches, a first one based on periodic exact-feature checks at the feature or output level, and another one based on the decoder confidence level. Internal evaluation supports both global and per-feature control, depending if we define a global metric or local ones.

5.2.2.1 Periodic Exact-Feature Check

The concept relies on periodic check windows, during which the same features are recomputed at a reference quality (highest Q or an exact baseline). In each check window, the approximate features are compared to the reference ones using a chosen metric, and the result is mapped to a target quality level Q through a LUT. If the metric stays above a target for consecutive checks, Q decreases, otherwise it increases.

When performing per-feature checks, a possible choice for the metric is the Pearson Correlation Coefficient (CC) over the check windows as shown in Equation 5.1.

$$CC(x,\hat{x}) = \frac{N\sum_{t=1}^{N} x_t\hat{x}_t - \left(\sum_{t=1}^{N} x_t\right)\left(\sum_{t=1}^{N} \hat{x}_t\right)}{\sqrt{\left(N\sum_{t=1}^{N} x_t^2 - (\sum_{t=1}^{N} x_t)^2\right)\left(N\sum_{t=1}^{N} \hat{x}_t^2 - (\sum_{t=1}^{N} \hat{x}_t)^2\right)}} \tag{5.1}$$

Where N is the check window length, x_t the approximate feature samples, and $\hat{x}_t$ the corresponding reference samples.

The AMNES system in [35] uses this equation to compute correlation using fixed-point arithmetic. The authors target an accelerator for data centers, where correlation is used for tasks such as feature selection in machine learning, and query optimization. AMNES implements a streaming correlation engine composed of an accumulator back-end that maintains running sums, sums of squares, and sums of products, and a front-end that uses these statistics to compute the correlation. Beyond Pearson CC, other metrics can be derived from the same running sums and products such as rank-based correlations, cosine similarity, and simple linear-regression statistics.

Other designs, introduced in [110], also compute the CC on Field Programmable Gate Array (FPGA), for signal template matching, where a measurement signal is correlated with a template. Two parametrizable modules are compared: a time-domain and a frequency domain approach. Both architectures support configurable parameters to balance ressource usage and performance.

In our system, we can use the same CC expression as an internal metric to compare approximate and reference features. Rather than maximizing throughput, correlation would be used as a FEP to monitor feature quality under strict power and area constraints.

Another possible approach is a decoding-level check. During defined check windows, the decoder is run twice: once with approximate features and once with reference features. An evaluation metric, such as the label mismatch rate for classification, is computed from the two outputs. Using a LUT, the output value is mapped to a global quality Q. This check can remain global or be class-aware. Class-specific errors can be used to control the quality for features that support the specific classes.

In [193], a teacher-student pair of Convolutional Neural Networks (CNNs) is used, with the low-power student active by default and the high-accuracy teacher enabled only at low-confidence transitions defined from a confusion matrix. In our approach, the confusion matrix between outputs, using the approximate and the reference features, could be used to increase or decrease the quality Q of feature extraction.

Despite the promising approaches, there remain a few points to be investigated. Future work should assess whether a low-power implementation of the full closed-loop mechanism is achievable within the energy budget of the targeted BCI, including the AQFE, and the controller. In particular, the cost of periodic exact-feature checks should be balanced with the frequency of updates so that overhead remains limited while sensitivity is preserved. Section 5.2.5 details a case-study that explains the feasibility constraints of such a system.

5.2.2.2 Decoder Confidence Level

Another internal FEP is the confidence level of the decoder. Instead of recomputing reference features, the controller monitors the confidence value at the output of the decoder. For probabilistic classifiers, this value can be obtained from the maximum class probability. These quantities are available for common BCI decoders such as regularized logistic regression, linear discriminant analysis, and CNNs, where the softmax scores

provide a confidence estimate. The confidence value is then mapped to a quality level Q through a LUT, similarly to periodic exact-feature checks. High-confidence windows allow decreasing Q to save energy, while low-confidence windows increase Q.

Confidence-aware decoding has already been explored in different BCI settings. For speech decoding from Electroencephalography (EEG), the approach described in [74] uses deep ensembles with uncertainty metrics and a selective accept/abstain rule driven by an accuracy-coverage trade-off. Similarly, the authors in [102] employ Bayesian neural networks and an adaptive uncertainty threshold to implement a reject option for motor-imagery tasks. In a BCI, used for Steady-State Visually Evoked Potentials (SSVEP) detection [190], the Bayesian confidence value is estimated from correlation-based features and each classification is accepted or rejected based on this value. In our context, similar confidence signals could be used to drive the feature quality.

5.2.3 External Sensing and User Feedback

In addition to internal indicators, external FEPs can be used to modulate the quality Q. They can be used as complementary inputs to the controller. As for internal FEPs, the corresponding metrics can be used to drive either a global quality level or per-feature adjustments. We detail two examples: a user satisfaction decoder that outputs the user's feedback, and external sensors which are used to adapt feature quality according to an external measurement.

5.2.3.1 User Satisfaction Decoder

An initial approach is to decode user satisfaction using a dedicated decoder. This additional model takes neural or behavioral signals as input, and decides whether the current BCI action matches the user's intention. The resulting score is used as an external FEP and mapped to a quality level Q through a LUT.

An auto-adaptive BCI framework developed at Clinatec, in [131], introduces a task-performance decoder that detects, from Electrocorticography (ECoG) activity in the sensorimotor cortex, whether the effector's action corresponds to the user's intention or not. The output is combined with that of the control decoder to generate labels and update the control decoder online, during unsupervised use. A cross-patient study in [39] extends this idea with a neural-response decoder and compares labeling strategies across several paradigms.

In our framework, such satisfaction or performance decoders could be reused as external FEPs to control feature quality. Persistant dissatisfaction would increase Q, while stable high satisfaction level would enable the value of Q to be lowered to reduce energy consumption. In practice, the energy cost of estimating satisfaction and running the controller should remain well below the additional cost of using the highest-quality features. Otherwise, the control loop would not provide a significant energy savings.

5.2.3.2 Secondary External Sensors

A second approach is to use external sensors as feedback signals. Such sensors provide information about kinematics or complementary measurements from the environment, independently of the feature extractor. Examples include Electromyography (EMG) electrodes to detect muscle activity, eye tracking using Electrooculography (EOG), or inertial

sensors to track head or limb movements. These sensors are often used to improve decoding accuracy by providing additional input or removing artefacts.

For instance, a hybrid BCI in [132] couples joystick data, EEG, and EMG for users with severe motor disabilities. In this setup, EMG is used as an additional control signal, providing hand movement detection. In [177], EEG and EMG patterns are jointly analyzed, as shown in Figure 5.3, and the EEG-EMG coupling is used as indicators of rehabilitation progress. A hybrid EEG/EOG system has been proposed in [153] to improve control of assistive devices such as hand exoskeletons. Inertial sensors have also been used to detect and suppress artefacts in mobile BCI systems. A method in [73] uses such sensors to remove motion artefacts from EEG in real-time.

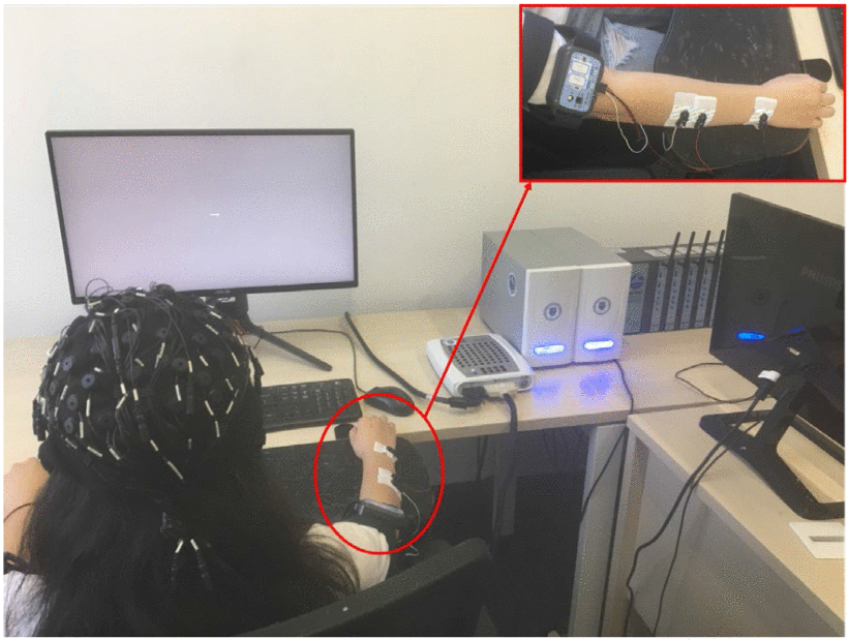


Figure 5.3: Experimental Setup with an EEG Headset and EMG Sensors, Adapted from [177].

In our framework, such secondary sensors could be reused as external FEPs to adapt the feature-extraction quality. Periods with strong muscle activity or large movements would trigger higher Q to preserve decoding performance, while stable and low-artifact conditions would allow lower Q for energy savings. External inputs could also be as simple as a user-operated knob that explicitly selects the quality Q.

5.2.4 Summary of Closed-loop Feedback Techniques

We summarize here the proposed feedback mechanisms. Table 5.1 classifies them by source (internal vs external) and control scope (global vs per-feature).

These different feedback mechanisms can be combined. For example, a fast internal FEP such as the decoder confidence (already computed during inference) can globally adjust Q, while slower, more costly, checks such feature-level correlation provide per-feature adjustments. Exploring how to combine these feedback sources with accuracy constraints and a fixed energy budget is left for future work.

Table 5.1: Summary of Closed-Loop Feedback by Source and Control Scope.

	Global	Per-Feature
Internal	Decoding-level checks of approximate vs exact features. Decoder confidence level.	Periodic exact-feature checks (CC or other similarity metric).
External	User satisfaction (through a decoder or an external button). External sensors for motion or artefact detection	Satisfaction or sensors with class-specific feature adjustment.

5.2.5 Feasibility of a BCI with Closed-loop Quality Control

The previous sections introduced several ways to select the feature quality Q using internal and external FEPs. We now consider a simple quantitative case study to assess under which conditions such a feedback loop can provide system-level energy savings.

We assume two quality levels for the feature extractor: a high-quality **HQ** and a low-quality **LQ**. Let $d \in [0, 1]$ denote the fraction of time (duty cycle) spent at high quality, and $(1 - d)$ the fraction spent at low quality. The controller consumes an average power P_{ctl}, to evaluate the FEPs and select the target quality Q. The average power of the AQFE and the controller is thus given in Equation 5.2.

$$P_{total} = d \times P_{HQ} + (1 - d) \times P_{LQ} + P_{ctl}. \tag{5.2}$$

We define the baseline as the configuration in which we always use the **HQ** configuration. The baseline power consumption is given by $P_{baseline} = P_{HQ}$.

We define the relative power saving s in Equation 5.3

$$s = \frac{P_{baseline} - P_{total}}{P_{baseline}} = 1 - \frac{P_{total}}{P_{baseline}} \tag{5.3}$$

By replacing P_{total} using Equation 5.2 and $P_{baseline} = P_{HQ}$, we obtain:

$$\begin{aligned} s &= 1 - \frac{d \times P_{HQ} + (1 - d) \times P_{LQ} + P_{ctl}}{P_{HQ}} \\ &= 1 - d - \frac{(1 - d) \times P_{LQ}}{P_{HQ}} - \frac{P_{ctl}}{P_{HQ}} \\ &= (1 - d)\Big(1 - \frac{P_{LQ}}{P_{HQ}}\Big) - \frac{P_{ctl}}{P_{HQ}} \end{aligned} \tag{5.4}$$

This expression shows that the relative saving s depends on three main factors. The first one $(1 - d)$ captures how often the system runs at low quality: the smaller the high-quality duty cycle d, the larger this factor and the higher the potential power reduction. Second, the ratio between low and high quality power, which appears as $\big(1 - P_{LQ}/P_{HQ}\big)$, is determined by the chosen quality levels and by the design of the feature extractor itself. If the two modes are too close in power, even an ideal controller cannot achieve significant power reduction. Third, the controller power P_{ctl} appears as a penalty term, and increasing it reduces s linearly. A high P_{ctl} can even lead to negative s, meaning

that the system with the feedback mechanism would consume more power than the initial system running permanently at **HQ**.

Requiring a non-negative saving $s \geq 0$ therefore imposes a simple constraint on d and P_{ctl}, given in Equation 5.5.

$$P_{ctl} \leq (1 - d)\big(P_{HQ} - P_{LQ}\big). \tag{5.5}$$

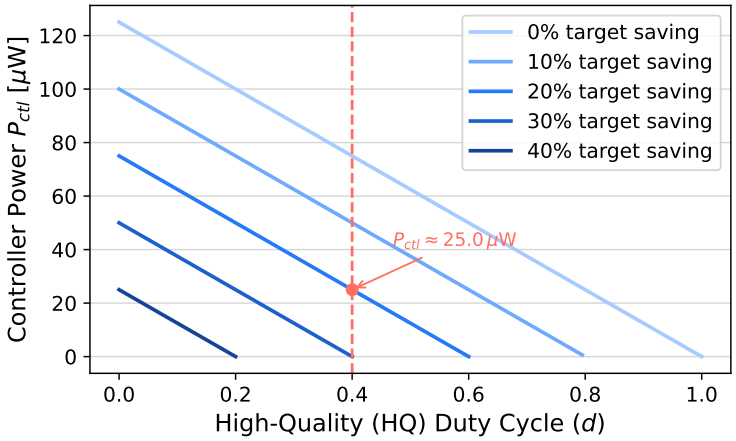


Figure 5.4: Controller Power P_{ctl} as a Function of the Duty Cycle d.

Figure 5.4 illustrates this by showing the maximum controller power P_{ctl} as a function of the duty cycle d for energy savings between 0% and 40%. The power values are those obtained in Chapter 4, corresponding to $P_{HQ} = 250\ \mu$W and $P_{LQ} = 125\ \mu$W

For a desired energy savings value, the admissible P_{ctl} decreases linearly as d increases, meaning that spending more time at **HQ** requires a proportionally lower power controller. Another way of looking at it is to observe that, for a given duty cycle, only duty cycles to the left of the corresponding curve are feasible. For example, at a duty cycle of $d = 0.4$, bounded by the dataset, the intersection with the 20% energy savings curve gives the upper bound on $P_{ctl} \approx 25\ \mu$W, as shown in Figure 5.4.

Conclusion 5.2.5

This simple case study shows that, with a given ratio between P_{HQ} and P_{LQ}, closed-loop quality control is only beneficial if the system can operate at **LQ** for a significant fraction of time and if the controller overhead remains small compared to the savings achieved by switching to lower quality.

5.3 Remaining Challenges

The previous sections introduced the building blocks of an adaptive-quality BCI: the AQFE, the quality parameter Q, and internal and external FEPs for closed-loop control. There remain multiple challenges before such a system can be deployed.

A first is the design of the control loop, which consists of choosing FEPs with the associated metrics. The mapping between these metrics and the quality levels must ensure stability, while meeting latency constraints. This includes choosing hysteresis thresholds, update rates, and deciding whether the control is local or global.

A second challenge is the hardware overhead of the feedback loop itself. Correlation engines, confidence or satisfaction decoders, and external sensors consume energy. For the AQFE to provide energy savings, the additional cost of computing FEPs and running the controller must remain significantly below the reduction achieved by operating at lower quality. This requires a low-power implementation of the feedback blocks and a careful choice of the period of the update cycle to balance between performance and energy.

A third challenge concerns robustness for long-term use, as neural signals evolve over time. The feedback loop should adapt to these changes through supervised or unsupervised recalibration. The complexity and efficiency of such mechanisms has to be taken into account.

Finally, evaluating such systems requires new clinical protocols. Offline analyses can estimate the impact of approximation on decoding accuracy and energy. However, closed-loop behavior can only be fully assessed in online experiments. Defining benchmarks, metrics, and experiments to compare quality-control strategies remains an open challenge for deploying such BCIs.

Chapter 6

Conclusions

This chapter concludes the thesis and summarizes the main results on low-power feature extraction for motor Brain-Computer Interfaces (BCIs). Starting from the gap identified between algorithmic requirements and embedded or implantable constraints, the work focused on reducing the compute cost, and thus the energy consumption, of time-frequency feature extraction. We proposed two algorithmic approaches for approximate wavelet transforms and a hardware implementation of an adaptive extractor, which preserve decoding performance. In this chapter, we summarize the main contributions, discuss their implications and limitations, and consider how this work can be applied to practical BCI systems.

6.1 Summary of Contributions

The goal of this thesis was to reduce the power consumption of brain-signal decoding for motor BCIs, while preserving high accuracy and meeting real-time constraints for embedded and future implantable systems. The work builds on the clinical BCI system used at Clinatec and uses a cross-layer approach, from feature extraction algorithms to their implementation in hardware.

Chapter 2 reviewed the state of the art in BCI decoding algorithms and hardware implementations, with a particular focus on feature extraction. From this study, we proposed a set of metrics that enable a fair comparison across heterogeneous systems, taking into account both decoding accuracy and circuit performance. The analysis showed that, in most embedded and implantable BCIs, the energy consumption is dominated by the feature-extraction pipeline rather than by the input data rate, highlighting feature extraction as the most compute-intensive stage. For the existing system developed at Clinatec, the algorithmic profiling in Appendix B.1.2 similarly identified the feature extraction stage as the main compute bottleneck, which motivated the thesis focus on reducing its complexity using a cross-layer hardware-software co-design approach.

In Chapter 3, we addressed the feature extraction bottleneck at the algorithmic level by introducing two approximate Continuous Wavelet Transform (CWT) methods, the Quantized Continuous Wavelet Transform (QCWT) and the Piecewise Linear Continuous Wavelet Transform (PLCWT). Both rely on piecewise approximations of the mother wavelet, reducing the number of arithmetic operations while keeping the time-frequency structure relevant for decoding. We analyzed the approximation error by computing the Correlation Coefficient (CC) between exact and approximate features on real Electro-

corticography (ECoG) data, and showed that the CC can serve as a proxy to evaluate decoding performance. Using operation-based energy models, we estimated that the proposed methods achieve up to 4× lower energy than the exact Fast Fourier Transform (FFT) method, for our use case, while limiting the decoding accuracy loss to within 2% of the baseline.

To confirm the energy savings estimation, we expanded the PLCWT method, in Chapter 4, into an Adaptive Quality Feature Extractor (AQFE), a hardware accelerator with dynamic control of the approximation level. The cross-layer approach follows a top-down path, from the algorithmic to the architectural, and finally to the circuit level. Post-layout characterization in 22 nm FDSOI process technology demonstrated that the AQFE can extract up to 960 features at 10 Hz, meeting the throughput requirements for motor decoding. The power per feature is reduced by at least 4× compared to prior digital feature extractors, while preserving decoding accuracy. Three architectures, each with a different number of lanes, are proposed to balance area, throughput and clock frequency. Depending on the level of parallelism, the silicon area is reduced by up to 2× compared to existing circuits. Runtime quality control in the AQFE provides an additional energy reduction of 2×, allowing the same hardware block to adapt according to task complexity and signal quality.

Building on this, Chapter 5 proposed a full BCI decoding system that places the AQFE between acquisition and decoding, optionally followed by a Complex-to-Real (C2R) post-processing block that computes energy features. A key concept introduced in this chapter is the use of Feature Evaluation Parameters (FEPs) to close the feedback loop. The quality of the features can be adjusted globally or per feature using internal indicators such as periodic exact-feature checks or decoder confidence, and external feedback such as user satisfaction or secondary sensors. Overall, this chapter explores how adaptive control of feature quality could be embedded in a BCI, providing a first step towards a future implementation in a low-power circuit.

Beyond these main axes, the thesis also contributed to the broader ecosystem around the target BCI platform. Embedded software optimizations enabled a 10× power reduction of the decoding and model-update pipeline on an embedded target (Raspberry Pi) compared to a previous portable PC implementation. Additional studies explored Phase-Amplitude Coupling (PAC) features, the design of a configurable C2R hardware block for feature energy computation, as well as an initial exploration of approximate multipliers. Taken together, these contributions provide new solutions for low-power time-frequency feature extractors for BCI.

6.2 Discussion and Limitations

The work presented promising results for low-power BCIs, but the results have some limitations. A first limitation is the scope of the evaluation. The approximate CWT methods were validated on motor tasks using ECoG data. These techniques remain to be validated for other BCI paradigms, such as speech decoding or Steady-State Visually Evoked Potentialss (SSVEPs) detection.

A second limitation is the separate study of algorithmic and hardware optimizations. The energy models in Chapter 3 calculated an approximate energy based on the number of operations, while the post-layout results in Chapter 4 quantify the power of the AQFE for a specific technology implementation with all optimizations enabled. We did not perform an incremental, ablation evaluation of the proposed design choices, which would

help separate the contribution of the PLCWT from that of the hardware optimizations. In addition, this work does not include a full System-on-Chip (SoC) with all components, and in a complete system, the interactions with other blocks such as decoders, controllers, and communication interfaces also affect the overall energy consumption.

A third limitation concerns the closed-loop control mechanisms introduced in Chapter 5. The proposed feedback strategies, based on internal and external FEPs, are conceptual and were not yet evaluated quantitatively. For the adaptive-quality mechanism to provide a net benefit, the energy cost of the feedback control system must remain significantly lower than the savings obtained by operating at reduced quality.

Finally, there are other constraints that are outside the scope of this thesis but are essential for making such a system usable in everyday, non-clinical settings. Safety requirements for medical implantable devices impose strict performance and robustness constraints, which are subject to regulatory processes. with approximate arithmetic, for example, this includes checks for component-error tolerance [130]. Dealing with these aspects will require collaboration with clinicians and regulatory specialists, while ultimately focusing on the user in the loop.

6.3 Perspectives and Future Work

This thesis took steps toward approximate, energy-aware feature extraction and adaptive-quality control in BCI systems. Beyond the short-term engineering extensions, it opens several research directions for longer-term perspectives on future BCIs.

In the short term, a first direction is to move from conceptual to concrete adaptive-quality feedback. This includes integrating simple feedback blocks and controllers alongside the AQFE, and characterizing their energy and latency overheads, preferably online, in real conditions. Such experiments would clarify how often the quality parameter can be updated, and how global and per-feature strategies compare. Furthermore, applying the same principles to other BCI paradigms would evaluate the robustness of approximate feature extraction methods.

In the longer term, the objective is to embed adaptive-quality feature extraction in complete systems suited for long-term use, and to extend the same co-design approach to other parts of the pipeline. One direction is to extend the cross-layer approach used in this thesis for feature extraction to the system level, by considering acquisition, feature extraction, and decoding as different levels of a single system whose energy budget can be dynamically adjusted across the pipeline depending on the task or the required performance. The decoding pipeline could be combined with on-chip learning to build BCIs that can maintain their accuracy while dynamically adjusting their energy usage. In practice, a model update algorithm would be ported to the circuit in order to calibrate the decoder and, optionally, the AQFE. More generally, this opens the door to bio-inspired BCIs that modulate their computational effort with task demands. It would operate by default at a low quality, and switch to a higher-cost, more accurate mode only when required, similarly to the brain which shifts from automatic to effortful processing when needed [69].

More generally, this opens the door to bio-inspired BCIs that modulate their computational effort as a function of task demand. The system could operate by default in a low-cost mode and switch to a higher-cost, more accurate mode only when needed, in line with the “fast” versus “effortful” modes discussed in [69].

Beyond algorithmic and hardware adaptation, a key perspective is to explicitly put the

user in the loop, so that the BCI and the user co-adapt over time. Ultimately, validating the benefits of approximate features and quality control will require online studies with a larger number of users to assess not only decoding performance and energy savings, but also usability in real-world assistive and rehabilitative scenarios.

6.4 Concluding Remarks

This thesis started from the observation that many BCI systems, especially those targeting motor rehabilitation, are constrained not only by decoding accuracy but also by power budgets. The introduction of an AQFE, based on approximate CWT, provides a concrete top-down approach for power optimization. Together with other progress in the BCI field, these approaches can help translate prototypes into systems suitable for home use.

The motivation for this work is to improve the quality of life of people living with spinal cord injury and motor disabilities. By addressing energy requirements for BCIs, this thesis contributes one step toward systems that can be deployed in home rehabilitation settings. It is my hope that the methods developed here will support future efforts to bring reliable, low-power BCIs closer to everyday use.

Bibliography

[1] [n. d.]. Armadillo C++ library for linear algebra & scientific computing. https://arma.sourceforge.net/.

[2] [n. d.]. Les lauréats des Trophées des XVIIèmes Assises de l'embarqué. https://assises.embedded-france.org/les-laureats-des-trophees-des-xviiemes-assises-de-lembarque/

[3] [n. d.]. OpenBLAS - An Optimized BLAS Library. https://www.openblas.net/.

[4] [n. d.]. Spinal Cord Injury. https://www.who.int/news-room/fact-sheets/detail/spinal-cord-injury.

[5] Mradul Agrawal, Sandeep Vidyashankar, and Ke Huang. 2016. On-chip implementation of ECoG signal data decoding in brain-computer interface. In *2016 IEEE 21st International Mixed-Signal Testing Workshop (IMSTW)*. https://doi.org/10.1109/IMS3TW.2016.7524225

[6] J. Montalvo Aguilar, J. Castillo, and D. Elias. 2015. EEG Signals Processing Based on Fractal Dimension Features and Classified by Neural Network and Support Vector Machine in Motor Imagery for a BCI. In *VI Latin American Congress on Biomedical Engineering CLAIB 2014, Paraná, Argentina 29, 30 & 31 October 2014*, Ariel Braidot and Alejandro Hadad (Eds.). Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-13117-7_157

[7] Faraz Akram, Seung Moo Han, and Tae-Seong Kim. 2015. An efficient word typing P300-BCI system using a modified T9 interface and random forest classifier. *Computers in Biology and Medicine* 56 (Jan. 2015). https://doi.org/10.1016/j.compbiomed.2014.10.021

[8] Ali H. Al-nuaimi, Emmanuel Jammeh, Lingfen Sun, and Emmanuel Ifeachor. 2017. Higuchi fractal dimension of the electroencephalogram as a biomarker for early detection of Alzheimer's disease. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. https://doi.org/10.1109/EMBC.2017.8037320

[9] Roxana Aldea and Daniela Tărniceriu. 2013. Estimating the hurst exponent in motor imagery-based brain computer interface. In *2013 7th Conference on Speech Technology and Human - Computer Dialogue (SpeD)*. https://doi.org/10.1109/SpeD.2013.6682656

[10] Daney Alex, Amitabh Yadav, Jieun Joo, Uisub Shin, Arshia Afzal, Jinhan Liu, Geoffrey Diehl, Alik Widge, and Mahsa Shoaran. 2025. A 32-Channel 196-μW Logarithmic SoC for Brain Network Connectivity Extraction and Adaptive Psychiatric Symptom Classification. In *2024 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*. IEEE. https://infoscience.epfl.ch/handle/20.500.14299/251241

[11] Hyochan An, Samuel R. Nason-Tomaszewski, Jongyup Lim, Kyumin Kwon, Matthew S. Willsey, Parag G. Patil, Hun-Seok Kim, Dennis Sylvester, Cynthia A. Chestek, and David Blaauw. 2022. A Power-Efficient Brain-Machine Interface System With a Sub-mw Feature Extraction and Decoding ASIC Demonstrated in Nonhuman Primates. *IEEE Transactions on Biomedical Circuits and Systems* 16, 3 (June 2022). https://doi.org/10.1109/TBCAS.2022.3175926

[12] Nana Arizumi and Tetiana Aksenova. 2019. Fast Continuous Wavelet Transform for Brain Computer Interface using piecewise polynomials. In *2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. https://doi.org/10.1109/ISSPIT47144.2019.9001739

[13] Lukas P. A. Arts and Egon L. van den Broek. 2022. The fast continuous wavelet transformation (fCWT) for real-time, high-quality, noise-resistant time–frequency analysis. *Nature Computational Science* 2, 1 (Jan. 2022). https://doi.org/10.1038/s43588-021-00183-z

[14] Onder Aydemir and Temel Kayikcioglu. 2011. Wavelet Transform Based Classification of Invasive Brain Computer Interface Data. *Radioengineering* 20 (April 2011).

[15] Onder Aydemir and Temel Kayikcioglu. 2014. Decision tree structure based classification of EEG signals recorded during two dimensional cursor movement imagery. *Journal of Neuroscience Methods* 229 (May 2014). https://doi.org/10.1016/j.jneumeth.2014.04.007

[16] Chanhee Bae, Seongjoo Lee, and Yunho Jung. 2022. High-Speed Continuous Wavelet Transform Processor for Vital Signal Measurement Using Frequency-Modulated Continuous Wave Radar. *Sensors* 22, 8 (Jan. 2022). https://doi.org/10.3390/s22083073

[17] S. Balakrishnama and A. Ganapathiraju. 1995. LINEAR DISCRIMINANT ANALYSIS - A BRIEF TUTORIAL. https://www.semanticscholar.org/paper/LINEAR-DISCRIMINANT-ANALYSIS-A-BRIEF-TUTORIAL-Balakrishnama-Ganapathiraju/163bb761cd7c5ca8c786bf4816a10308f1396ffb

[18] Ulvi Baspinar and Yahya Taştan. 2025. Enhancing Motor Imagery EEG Classification in Brain–Computer Interfaces via TransformerNet. *IEEE Access* 13 (2025). https://doi.org/10.1109/ACCESS.2025.3594083

[19] Joan Belo, Maureen Clerc, and Daniele Schön. 2021. EEG-Based Auditory Attention Detection and Its Possible Future Applications for Passive BCI. *Frontiers in Computer Science* 3 (April 2021). https://doi.org/10.3389/fcomp.2021.661178

[20] Alim Louis Benabid, Thomas Costecalde, Andrey Eliseyev, Guillaume Charvet, Alexandre Verney, Serpil Karakas, Michael Foerster, Aurélien Lambert, Boris Morinière, Neil Abroug, Marie-Caroline Schaeffer, Alexandre Moly, Fabien Sauter-Starace, David Ratel, Cecile Moro, Napoleon Torres-Martinez, Lilia Langar, Manuela Oddoux, Mircea Polosan, Stephane Pezzani, Vincent Auboiroux, Tetiana Aksenova, Corinne Mestais, and Stephan Chabardes. 2019. An exoskeleton controlled by an epidural wireless brain–machine interface in a tetraplegic patient: a proof-of-concept demonstration. *The Lancet Neurology* 18, 12 (Dec. 2019). https://doi.org/10.1016/S1474-4422(19)30321-7

[21] B. Blankertz, K.-R. Muller, D.J. Krusienski, G. Schalk, J.R. Wolpaw, A. Schlogl, G. Pfurtscheller, Jd.R. Millan, M. Schroder, and N. Birbaumer. 2006. The BCI competition III: validating alternative approaches to actual BCI problems. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 14, 2 (June 2006). https://doi.org/10.1109/TNSRE.2006.875642

[22] Florent Bocquelet, Thomas Hueber, Laurent Girin, Christophe Savariaux, and Blaise Yvert. 2016. Real-Time Control of an Articulatory-Based Speech Synthesizer for Brain Computer Interfaces. *PLOS Computational Biology* 12, 11 (Nov. 2016), e1005119. https://doi.org/10.1371/journal.pcbi.1005119

[23] Fabio Boi, Timoleon Moraitis, Vito De Feo, Francesco Diotalevi, Chiara Bartolozzi, Giacomo Indiveri, and Alessandro Vato. 2016. A Bidirectional Brain-Machine Interface Featuring a Neuromorphic Hardware Decoder. *Frontiers in Neuroscience* 10 (Dec. 2016). https://doi.org/10.3389/fnins.2016.00563

[24] Alberto Bosio, Daniel Ménard, and Olivier Sentieys (Eds.). 2022. *Approximate Computing Techniques: From Component- to Application-Level*. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-030-94705-7

[25] V. Bostanov. 2004. BCI competition 2003-data sets Ib and IIb: feature extraction from event-related brain potentials with the continuous wavelet transform and the t-value scalogram. *IEEE Transactions on Biomedical Engineering* 51, 6 (June 2004). https://doi.org/10.1109/TBME.2004.826702

[26] Rasmus Bro. 1997. PARAFAC. Tutorial and applications. *Chemometrics and Intelligent Laboratory Systems* 38, 2 (Oct. 1997). https://doi.org/10.1016/S0169-7439(97)00032-4

[27] Emmanuel Candes, Mark Rudelson, Terence Tao, and Roman Vershynin. 2005. Error correction via linear programming. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*. IEEE, Pittsburgh, PA. https://doi.org/10.1109/SFCS.2005.5464411

[28] Obed Carrera-León, Juan Manuel Ramirez, Vicente Alarcon-Aquino, Mary Baker, David D'Croz-Baron, and Pilar Gomez-Gil. 2012. A motor imagery BCI experiment using wavelet analysis and spatial patterns feature extraction. In *2012 Workshop on Engineering Applications*. https://doi.org/10.1109/WEA.2012.6220084

[29] R. Carter, J. Mazurier, L. Pirro, J-U. Sachse, P. Baars, J. Faul, C. Grass, G. Grasshoff, P. Javorka, T. Kammler, A. Preusse, S. Nielsen, T. Heller, J. Schmidt,

H. Niebojewski, P-Y. Chou, E. Smith, E. Erben, C. Metze, C. Bao, Y. Andee, I. Aydin, S. Morvan, J. Bernard, E. Bourjot, T. Feudel, D. Harame, R. Nelluri, H.-J. Thees, L. M-Meskamp, J. Kluth, R. Mulfinger, M. Rashed, R. Taylor, C. Weintraub, J. Hoentschel, M. Vinet, J. Schaeffer, and B. Rice. 2016. 22nm FDSOI technology for emerging mobile, Internet-of-Things, and RF applications. In *2016 IEEE International Electron Devices Meeting (IEDM)*. https://doi.org/10.1109/IEDM.2016.7838029

[30] Alireza Chamanzar, Mahdi Shabany, Alireza Malekmohammadi, and Sara Mohammadinejad. 2017. Efficient Hardware Implementation of Real-Time Low-Power Movement Intention Detector System Using FFT and Adaptive Wavelet Transform. *IEEE Transactions on Biomedical Circuits and Systems* 11, 3 (June 2017). https://doi.org/10.1109/TBCAS.2017.2669911

[31] Rajdeep Chatterjee and Tathagata Bandyopadhyay. 2016. EEG Based Motor Imagery Classification Using SVM and MLP. In *2016 2nd International Conference on Computational Intelligence and Networks (CINE)*. https://doi.org/10.1109/CINE.2016.22

[32] Tung-Chien Chen, Tsung-Hsueh Lee, Yu-Hsin Chen, Tsung-Chuan Ma, Tzu-Der Chuang, Chien-Jung Chou, Chung-Hsing Yang, Tsung-Hsien Lin, and Liang-Gee Chen. 2010. 1.4µW/channel 16-channel EEG/ECoG processor for smart brain sensor SoC. In *2010 Symposium on VLSI Circuits*. https://doi.org/10.1109/VLSIC.2010.5560258

[33] Tung-Chien Chen, Wentai Liu, and Liang-Gee Chen. 2008. VLSI architecture of leading eigenvector generation for on-chip principal component analysis spike sorting system. In *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. https://doi.org/10.1109/IEMBS.2008.4649882

[34] Yi Chen, Enyi Yao, and Arindam Basu. 2015. A 128 channel 290 GMACs/W machine learning based co-processor for intention decoding in brain machine interfaces. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. https://doi.org/10.1109/ISCAS.2015.7169319

[35] Monica Chiosa, Thomas B. Preußer, Michaela Blott, and Gustavo Alonso. 2023. AMNES: Accelerating the Computation of Data Correlation Using FPGAs. *Proc. VLDB Endow.* 16, 13 (Sept. 2023). https://doi.org/10.14778/3625054.3625056

[36] Adelson Chua, Michael I. Jordan, and Rikky Muller. 2022. SOUL: An Energy-Efficient Unsupervised Online Learning Seizure Detection Classifier. *IEEE Journal of Solid-State Circuits* 57, 8 (Aug. 2022). https://doi.org/10.1109/JSSC.2022.3172231

[37] Annalisa Colucci, Mareike Vermehren, Alessia Cavallo, Cornelius Angerhöfer, Niels Peekhaus, Loredana Zollo, Won-Seok Kim, Nam-Jong Paik, and Surjo R. Soekadar. 2022. Brain–Computer Interface-Controlled Exoskeletons in Clinical Neurorehabilitation: Ready or Not? *Neurorehabilitation and Neural Repair* 36, 12 (Dec. 2022). https://doi.org/10.1177/15459683221138751

[38] Germund Dahlquist. 1956. Convergence and Stability in the Numerical Integration of Ordinary Differential Equations. *Math. Scand.* 4, 1 (1956). https://www.jstor.org/stable/24490010

[39] Hugo Lafaye de Micheaux, Felix Martel, Fabien Sauter, Guillaume Charvet, Henri Lorach, and Tetiana Aksenova. 2024. Auto-adaptation of ecog-based motor bci using neural response decoder: a cross-patient study. In *9th Graz Brain-Computer Interface Conference 2024*. https://cea.hal.science/cea-04709829/

[40] Taisija Demchenko and Milena Korostenskaja. 2022. Training CNN to Detect Motor Imagery in ECoG Data Recorded During Dreaming. In *HCI International 2022 - Late Breaking Papers. Multimodality in Advanced Interaction Environments (Lecture note2s in Computer Science)*, Masaaki Kurosu, Sakae Yamamoto, Hirohiko Mori, Dylan D. Schmorrow, Cali M. Fidopiastis, Norbert A. Streitz, and Shin'ichi Konomi (Eds.). Springer Nature Switzerland, Cham. https://doi.org/10.1007/978-3-031-17618-0_28

[41] Enzeng Dong, Changhai Li, Liting Li, Shengzhi Du, Abdelkader Nasreddine Belkacem, and Chao Chen. 2017. Classification of multi-class motor imagery with a novel hierarchical SVM algorithm for brain–computer interfaces. *Medical & Biological Engineering & Computing* 55, 10 (Oct. 2017). https://doi.org/10.1007/s11517-017-1611-4

[42] S.A. Dyer and J.S. Dyer. 2001. Cubic-spline interpolation. 1. *IEEE Instrumentation & Measurement Magazine* 4, 1 (March 2001). https://doi.org/10.1109/5289.911175

[43] Andrey Eliseyev, Vincent Auboiroux, Thomas Costecalde, Lilia Langar, Guillaume Charvet, Corinne Mestais, Tetiana Aksenova, and Alim-Louis Benabid. 2017. Recursive Exponentially Weighted N-way Partial Least Squares Regression with Recursive-Validation of Hyper-Parameters in Brain-Computer Interface Applications. *Scientific Reports* 7, 1 (Nov. 2017). https://doi.org/10.1038/s41598-017-16579-9

[44] Darjn Esposito, Davide De Caro, Ettore Napoli, Nicola Petra, and Antonio G. M. Strollo. 2017. On the use of approximate adders in carry-save multiplier-accumulators. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. https://doi.org/10.1109/ISCAS.2017.8050437

[45] Adrian Evans, Victor Roux-Sibillon, Joe Saad, Ivan Miro-Panades, Tetiana Aksenova, and Lorena Anghel. 2025. Enabling a Portable Brain Computer Interface for Rehabilitation of Spinal Cord Injuries. In *2025 Design, Automation & Test in Europe Conference (DATE)*. https://doi.org/10.23919/DATE64628.2025.10993056

[46] Remi Flamary and Alain Rakotomamonjy. 2012. Decoding Finger Movements from ECoG Signals Using Switching Linear Models. *Frontiers in Neuroscience* 6 (March 2012). https://doi.org/10.3389/fnins.2012.00029

[47] Mario Garrido. 2022. A Survey on Pipelined FFT Hardware Architectures. *Journal of Signal Processing Systems* 94, 11 (Nov. 2022). https://doi.org/10.1007/s11265-021-01655-1

[48] Moritz Gerster, Gunnar Waterstraat, Vladimir Litvak, Klaus Lehnertz, Alfons Schnitzler, Esther Florin, Gabriel Curio, and Vadim Nikulin. 2022. Separating Neural Oscillations from Aperiodic 1/f Activity: Challenges and Recommendations. *Neuroinformatics* 20, 4 (Oct. 2022). https://doi.org/10.1007/s12021-022-09581-8

[49] Daniela Gifu. 2021. The Use of Decision Trees for Analysis of the Epilepsy. *Procedia Computer Science* 192 (Jan. 2021). https://doi.org/10.1016/j.procs.2021.09.055

[50] A. Grossmann and J. Morlet. 1984. Decomposition of Hardy Functions into Square Integrable Wavelets of Constant Shape. *SIAM Journal on Mathematical Analysis* 15, 4 (July 1984). https://doi.org/10.1137/0515056

[51] Mahdiyeh Hajibabazadeh and Vahid Azimirad. 2014. Brain-robot interface: Distinguishing left and right hand EEG signals through SVM. In *2014 Second RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*. https://doi.org/10.1109/ICRoM.2014.6991004

[52] Jiri Hammer, Jörg Fischer, Johanna Ruescher, Andreas Schulze-Bonhage, Ad Aertsen, and Tonio Ball. 2013. The role of ECoG magnitude and phase in decoding position, velocity, and acceleration during continuous motor behavior. *Frontiers in Neuroscience* 7 (2013). https://www.frontiersin.org/articles/10.3389/fnins.2013.00200

[53] Richard A. Harshman and Margaret E. Lundy. 1994. PARAFAC: Parallel factor analysis. *Computational Statistics & Data Analysis* 18, 1 (Aug. 1994). https://doi.org/10.1016/0167-9473(94)90132-5

[54] Shousheng He and M. Torkelson. 1998. Design and implementation of a 1024-point pipeline FFT processor. In *Proceedings of the IEEE 1998 Custom Integrated Circuits Conference (Cat. No.98CH36143)*. https://doi.org/10.1109/CICC.1998.694922

[55] Christopher E. Heil and David F. Walnut. 1989. Continuous and Discrete Wavelet Transforms. *SIAM Rev.* 31, 4 (Dec. 1989). https://doi.org/10.1137/1031129

[56] Amin Hekmatmanesh, Huapeng Wu, Fatemeh Jamaloo, Ming Li, and Heikki Handroos. 2020. A combination of CSP-based method with soft margin SVM classifier and generalized RBF kernel for imagery-based brain computer interface applications. *Multimedia Tools and Applications* 79, 25 (July 2020). https://doi.org/10.1007/s11042-020-08675-2

[57] Salah Helmy, Tarik Al-ani, Yskandar Hamam, and Essam El-madbouly. 2008. P300 Based Brain-Computer Interface Using Hidden Markov Models. In *2008 International Conference on Intelligent Sensors, Sensor Networks and Information Processing*. https://doi.org/10.1109/ISSNIP.2008.4761974

[58] Christian Herff, Lorenz Diener, Miguel Angrick, Emily Mugler, Matthew C. Tate, Matthew A. Goldrick, Dean J. Krusienski, Marc W. Slutzky, and Tanja Schultz. 2019. Generating Natural, Intelligible Speech From Brain Activity in Motor,

Premotor, and Inferior Frontal Cortices. *Frontiers in Neuroscience* 13 (2019). https://www.frontiersin.org/articles/10.3389/fnins.2019.01267

[59] Ulrich Hoffmann, Jean-Marc Vesin, Touradj Ebrahimi, and Karin Diserens. 2008. An efficient P300-based brain–computer interface for disabled subjects. *Journal of Neuroscience Methods* 167, 1 (Jan. 2008). https://doi.org/10.1016/j.jneumeth.2007.03.005

[60] Mark Horowitz. 2014. 1.1 Computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. https://doi.org/10.1109/ISSCC.2014.6757323

[61] Essam H. Houssein, Asmaa Hammad, and Abdelmgeid A. Ali. 2022. Human emotion recognition from EEG-based brain–computer interface using machine learning: a comprehensive review. *Neural Computing and Applications* 34, 15 (Aug. 2022). https://doi.org/10.1007/s00521-022-07292-4

[62] Md. Redwan Islam, Umme Fatema, Mohammed Imamul Hassan Bhuiyan, and Syed Khairul Bashar. 2016. Classification of electrocorticography based motor imagery movements using continuous wavelet transform. In *2016 IEEE Students' Technology Symposium (TechSym)*. https://doi.org/10.1109/TechSym.2016.7872647

[63] Ritwik Jain, Prakhar Jaiman, and Veeky Baths. 2023. Feature Engineering for an Efficient Motor Related EcoG BCI System. https://doi.org/10.1101/2023.04.01.535201

[64] Amin Jarrah and Mohsin M. Jamali. 2014. Optimized FPGA based implementation of discrete wavelet transform. In *2014 48th Asilomar Conference on Signals, Systems and Computers*. https://doi.org/10.1109/ACSSC.2014.7094786

[65] Honglan Jiang, Cong Liu, Leibo Liu, Fabrizio Lombardi, and Jie Han. 2017. A Review, Classification, and Comparative Evaluation of Approximate Arithmetic Circuits. *ACM Journal on Emerging Technologies in Computing Systems* 13 (Aug. 2017). https://doi.org/10.1145/3094124

[66] Honglan Jiang, Cong Liu, Naman Maheshwari, Fabrizio Lombardi, and Jie Han. 2016. A comparative evaluation of approximate multipliers. In *2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*. https://doi.org/10.1145/2950067.2950068

[67] Tianxiao Jiang, Tao Jiang, Taylor Wang, Shanshan Mei, Qingzhu Liu, Yunlin Li, Xiaofei Wang, Sujit Prabhu, Zhiyi Sha, and Nuri F. Ince. 2017. Characterization and Decoding the Spatial Patterns of Hand Extension/Flexion using High-Density ECoG. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 25, 4 (April 2017). https://doi.org/10.1109/TNSRE.2016.2647255

[68] H. Johnson and C. Burrus. 1984. An in-order, in-place radix-2 FFT. In *ICASSP '84. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 9. https://doi.org/10.1109/ICASSP.1984.1172660

[69] Daniel Kahneman. 2011. *Thinking, fast and slow* (1st ed ed.). Farrar, Straus and Giroux, New York.

[70] Evelyn Karikari and Konstantin A. Koshechkin. 2023. Review on brain-computer interface technologies in healthcare. *Biophysical Reviews* 15, 5 (Oct. 2023). https://doi.org/10.1007/s12551-023-01138-6

[71] Chea-Yau Kee, S. G. Ponnambalam, and Chu-Kiong Loo. 2017. Binary and multi-class motor imagery using Renyi entropy for feature extraction. *Neural Computing and Applications* 28, 8 (Aug. 2017). https://doi.org/10.1007/s00521-016-2178-y

[72] Rabia Avais Khan, Nasir Rashid, Muhammad Shahzaib, Umar Farooq Malik, Arshia Arif, Javaid Iqbal, Mubasher Saleem, Umar Shahbaz Khan, and Mohsin Tiwana. 2023. A novel framework for classification of two-class motor imagery EEG signals using logistic regression classification algorithm. *PLOS ONE* 18, 9 (Sept. 2023). https://doi.org/10.1371/journal.pone.0276133

[73] Byung Hyung Kim, Jinsung Chun, and Sungho Jo. 2015. Dynamic motion artifact removal using inertial sensors for mobile BCI. In *2015 7th International IEEE/EMBS Conference on Neural Engineering (NER)*. https://doi.org/10.1109/NER.2015.7146554

[74] Soowon Kim, Byung-Kwan Ko, and Seo-Hyun Lee. 2025. Confidence-Aware Neural Decoding of Overt Speech from EEG: Toward Robust Brain-Computer Interfaces. https://doi.org/10.48550/arXiv.2511.07890

[75] Hou-Jen Ko and Shen-Fu Hsiao. 2011. Design and Application of Faithfully Rounded and Truncated Multipliers With Combined Deletion, Reduction, Truncation, and Rounding. *IEEE Transactions on Circuits and Systems II: Express Briefs* 58, 5 (May 2011). https://doi.org/10.1109/TCSII.2011.2148970

[76] Wonjun Ko, Jeeseok Yoon, Eunsong Kang, Eunji Jun, Jun-Sik Choi, and Heung-Il Suk. 2018. Deep recurrent spatio-temporal neural network for motor imagery based BCI. In *2018 6th International Conference on Brain-Computer Interface (BCI)*. https://doi.org/10.1109/IWW-BCI.2018.8311535

[77] Christelle Larzabal, Stéphane Bonnet, Thomas Costecalde, Vincent Auboiroux, Guillaume Charvet, Stéphan Chabardes, Tetiana Aksenova, and Fabien Sauter-Starace. 2021. Long-term stability of the chronic epidural wireless recorder WIMAGINE in tetraplegic patients. *Journal of Neural Engineering* 18, 5 (Sept. 2021). https://doi.org/10.1088/1741-2552/ac2003

[78] Vernon J. Lawhern, Amelia J. Solon, Nicholas R. Waytowich, Stephen M. Gordon, Chou P. Hung, and Brent J. Lance. 2018. EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces. *Journal of Neural Engineering* 15, 5 (July 2018). https://doi.org/10.1088/1741-2552/aace8c

[79] Hyeon Kyu Lee and Young-Seok Choi. 2019. Application of Continuous Wavelet Transform and Convolutional Neural Network in Decoding Motor Imagery Brain-Computer Interface. *Entropy* 21, 12 (Dec. 2019). https://doi.org/10.3390/e21121199

[80] Jiajun Li, Yu Qi, and Gang Pan. 2023. Phase-amplitude coupling-based adaptive filters for neural signal decoding. *Frontiers in Neuroscience* 17 (2023). https://www.frontiersin.org/articles/10.3389/fnins.2023.1153568

[81] Mingai Li, Jinfu Yang, Dongmei Hao, and Songmin Jia. 2009. ECoG recognition of motor imagery based on SVM Ensemble. In *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. https://doi.org/10.1109/ROBIO.2009.5420544

[82] Yang Li, Xu-Dong Wang, Mei-Lin Luo, Ke Li, Xiao-Feng Yang, and Qi Guo. 2018. Epileptic Seizure Classification of EEGs Using Time–Frequency Analysis Based Multiscale Radial Basis Functions. *IEEE Journal of Biomedical and Health Informatics* 22, 2 (March 2018). https://doi.org/10.1109/JBHI.2017.2654479

[83] Zheng Li, Joseph E. O'Doherty, Timothy L. Hanson, Mikhail A. Lebedev, Craig S. Henriquez, and Miguel A. L. Nicolelis. 2009. Unscented Kalman Filter for Brain-Machine Interfaces. *PLOS ONE* 4, 7 (2009). https://doi.org/10.1371/journal.pone.0006243

[84] Jongyup Lim, Eunseong Moon, Michael Barrow, Samuel R. Nason, Paras R. Patel, Parag G. Patil, Sechang Oh, Inhee Lee, Hun-Seok Kim, Dennis Sylvester, David Blaauw, Cynthia A. Chestek, Jamie Phillips, and Taekwang Jang. 2020. 26.9 A 0.19×0.17mm2 Wireless Neural Recording IC for Motor Prediction with Near-Infrared-Based Power and Data Telemetry. In *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*. https://doi.org/10.1109/ISSCC19947.2020.9063005

[85] Xiaokun Liu, Chunlai Zhang, Zheng Ji, Yi Ma, Xiaoming Shang, Qi Zhang, Wencheng Zheng, Xia Li, Jun Gao, Ruofan Wang, Jiang Wang, and Haitao Yu. 2016. Multiple characteristics analysis of Alzheimer's electroencephalogram by power spectral density and Lempel–Ziv complexity. *Cognitive Neurodynamics* 10, 2 (April 2016). https://doi.org/10.1007/s11571-015-9367-8

[86] Xilin Liu, Milin Zhang, Andrew G. Richardson, Timothy H. Lucas, and Jan Van der Spiegel. 2017. Design of a Closed-Loop, Bidirectional Brain Machine Interface System With Energy Efficient Neural Feature Extraction and PID Control. *IEEE Transactions on Biomedical Circuits and Systems* 11, 4 (Aug. 2017). https://doi.org/10.1109/TBCAS.2016.2622738

[87] Vladislav Lomtev, Alexander Kovalev, and Alexey Timchenko. 2023. FingerFlex: Inferring Finger Trajectories from ECoG signals. https://doi.org/10.48550/arXiv.2211.01960

[88] Henri Lorach, Andrea Galvez, Valeria Spagnolo, Felix Martel, Serpil Karakas, Nadine Intering, Molywan Vat, Olivier Faivre, Cathal Harte, Salif Komi, Jimmy Ravier, Thibault Collin, Laure Coquoz, Icare Sakr, Edeny Baaklini, Sergio Daniel Hernandez-Charpak, Gregory Dumont, Rik Buschman, Nicholas Buse, Tim Denison, Ilse van Nes, Leonie Asboth, Anne Watrin, Lucas Struber, Fabien Sauter-Starace, Lilia Langar, Vincent Auboiroux, Stefano Carda, Stephan Chabardes, Tetiana Aksenova, Robin Demesmaeker, Guillaume Charvet, Jocelyne Bloch,

and Grégoire Courtine. 2023. Walking naturally after spinal cord injury using a brain–spine interface. *Nature* (May 2023). https://doi.org/10.1038/s41586-023-06094-5

[89] Tom Lyche, Carla Manni, and Hendrik Speleers. 2018. Foundations of Spline Theory: B-Splines, Spline Approximation, and Hierarchical Refinement. In *Splines and PDEs: From Approximation Theory to Numerical Linear Algebra: Cetraro, Italy 2017*, Angela Kunoth, Tom Lyche, Giancarlo Sangalli, Stefano Serra-Capizzano, Tom Lyche, Carla Manni, and Hendrik Speleers (Eds.). Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-94911-6_1

[90] Xunguang Ma, Wenkai Zheng, Zujian Peng, and Jimin Yang. 2019. FPGA-Based Rapid Electroencephalography Signal Classification System. In *2019 IEEE 11th International Conference on Advanced Infocomm Technology (ICAIT)*. https://doi.org/10.1109/ICAIT.2019.8935935

[91] Juliano Machado and Alexandre Balbinot. 2014. Executed Movement Using EEG Signals through a Naive Bayes Classifier. *Micromachines* 5, 4 (Dec. 2014). https://doi.org/10.3390/mi5041082

[92] Kei Majima, Takeshi Matsuo, Keisuke Kawasaki, Kensuke Kawai, Nobuhito Saito, Isao Hasegawa, and Yukiyasu Kamitani. 2014. Decoding visual object categories from temporal correlations of ECoG signals. *NeuroImage* 90 (April 2014). https://doi.org/10.1016/j.neuroimage.2013.12.020

[93] Omid Malekzadeh-Arasteh, Haoran Pu, Jeffrey Lim, Charles Yu Liu, An H. Do, Zoran Nenadic, and Payam Heydari. 2020. An Energy-Efficient CMOS Dual-Mode Array Architecture for High-Density ECoG-Based Brain-Machine Interfaces. *IEEE Transactions on Biomedical Circuits and Systems* 14, 2 (April 2020). https://doi.org/10.1109/TBCAS.2019.2963302

[94] Ravikiran Mane, Tushar Chouhan, and Cuntai Guan. 2020. BCI for stroke rehabilitation: motor and beyond. *Journal of Neural Engineering* 17, 4 (Aug. 2020). https://doi.org/10.1088/1741-2552/aba162

[95] Morgane Marzulli, Alexandre Bleuzé, Joe Saad, Felix Martel, Philippe Ciuciu, Tetiana Aksenova, and Lucas Struber. 2025. Classifying mental motor tasks from chronic ECoG-BCI recordings using phase-amplitude coupling features. *Frontiers in Human Neuroscience* 19 (March 2025). https://doi.org/10.3389/fnhum.2025.1521491

[96] Kojiro Matsushita, Masayuki Hirata, Takafumi Suzuki, Hiroshi Ando, Takeshi Yoshida, Yuki Ota, Fumihiro Sato, Shayne Morris, Hisato Sugata, Tetsu Goto, Takufumi Yanagisawa, and Toshiki Yoshimine. 2018. A Fully Implantable Wireless ECoG 128-Channel Recording Device for Human Brain–Machine Interfaces: W-HERBS. *Frontiers in Neuroscience* 12 (2018). https://www.frontiersin.org/articles/10.3389/fnins.2018.00511

[97] Edwin M. Maynard, Craig T. Nordhausen, and Richard A. Normann. 1997. The Utah Intracortical Electrode Array: A recording structure for potential brain-computer interfaces. *Electroencephalography and Clinical Neurophysiology* 102, 3 (March 1997). https://doi.org/10.1016/S0013-4694(96)95176-0

[98] Dennis J. McFarland, Laurie A. Miner, Theresa M. Vaughan, and Jonathan R. Wolpaw. 2000. Mu and Beta Rhythm Topographies During Motor Imagery and Actual Movements. *Brain Topography* 12, 3 (March 2000). https://doi.org/10.1023/A:1023437823106

[99] Corinne S. Mestais, Guillaume Charvet, Fabien Sauter-Starace, Michael Foerster, David Ratel, and Alim Louis Benabid. 2015. WIMAGINE: Wireless 64-Channel ECoG Recording Implant for Long Term Clinical Applications. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 23, 1 (Jan. 2015). https://doi.org/10.1109/TNSRE.2014.2333541

[100] Sean L. Metzger, Jessie R. Liu, David A. Moses, Maximilian E. Dougherty, Margaret P. Seaton, Kaylo T. Littlejohn, Josh Chartier, Gopala K. Anumanchipalli, Adelyn Tu-Chan, Karunesh Ganguly, and Edward F. Chang. 2022. Generalizable spelling using a speech neuroprosthesis in an individual with severe limb and vocal paralysis. *Nature Communications* 13, 1 (Nov. 2022). https://doi.org/10.1038/s41467-022-33611-3

[101] Aleksandar Miladinović, Miloš Ajčević, Piero Paolo Battaglini, Giulia Silveri, Gaia Ciacchi, Giulietta Morra, Joanna Jarmolowska, and Agostino Accardo. 2020. Slow Cortical Potential BCI Classification Using Sparse Variational Bayesian Logistic Regression with Automatic Relevance Determination. In *XV Mediterranean Conference on Medical and Biological Engineering and Computing – MEDICON 2019*, Jorge Henriques, Nuno Neves, and Paulo de Carvalho (Eds.). Springer International Publishing, Cham. https://doi.org/10.1007/978-3-030-31635-8_225

[102] Daily Milanés-Hermosilla, Rafael Trujillo-Codorniú, Saddid Lamar-Carbonell, Roberto Sagaró-Zamora, Jorge Jadid Tamayo-Pacheco, John Jairo Villarejo-Mayor, and Denis Delisle-Rodriguez. 2023. Robust Motor Imagery Tasks Classification Approach Using Bayesian Neural Network. *Sensors* 23, 2 (Jan. 2023). https://doi.org/10.3390/s23020703

[103] Kai J. Miller. 2019. A library of human electrocorticographic data and analyses. *Nature Human Behaviour* 3, 11 (Nov. 2019). https://doi.org/10.1038/s41562-019-0678-3

[104] Kai J. Miller, Dora Hermes, Christopher J. Honey, Adam O. Hebb, Nick F. Ramsey, Robert T. Knight, Jeffrey G. Ojemann, and Eberhard E. Fetz. 2012. Human Motor Cortical Activity Is Selectively Phase-Entrained on Underlying Rhythms. *PLOS Computational Biology* 8, 9 (Sept. 2012). https://doi.org/10.1371/journal.pcbi.1002655

[105] Alexandre Moly, Alexandre Aksenov, Félix Martel, and Tetiana Aksenova. 2023. Online adaptive group-wise sparse Penalized Recursive Exponentially Weighted N-way Partial Least Square for epidural intracranial BCI. *Frontiers in Human Neuroscience* 17 (March 2023). https://doi.org/10.3389/fnhum.2023.1075666

[106] Alexandre Moly, Thomas Costecalde, Félix Martel, Matthieu Martin, Christelle Larzabal, Serpil Karakas, Alexandre Verney, Guillaume Charvet, Stephan Chabardes, Alim Louis Benabid, and Tetiana Aksenova. 2022. An adaptive

closed-loop ECoG decoder for long-term and stable bimanual control of an exoskeleton by a tetraplegic. *Journal of Neural Engineering* 19, 2 (March 2022). https://doi.org/10.1088/1741-2552/ac59a0

[107] Eiji Morifuji, Dinesh Patil, Mark Horowitz, and Yoshio Nishi. 2007. Power Optimization for SRAM and Its Scaling. *IEEE Transactions on Electron Devices* 54, 4 (April 2007). https://doi.org/10.1109/TED.2007.891869

[108] Vojtech Mrazek, Radek Hrbacek, Zdenek Vasicek, and Lukas Sekanina. 2017. EvoApprox8b: Library of Approximate Adders and Multipliers for Circuit Design and Benchmarking of Approximation Methods. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. https://doi.org/10.23919/DATE.2017.7926993

[109] Vojtech Mrazek, Zdenek Vasicek, Lukas Sekanina, Honglan Jiang, and Jie Han. 2018. Scalable Construction of Approximate Multipliers With Formally Guaranteed Worst Case Error. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26, 11 (Nov. 2018). https://doi.org/10.1109/TVLSI.2018.2856362

[110] Krzysztof Mroczek. 2025. FPGA Implementation of Normalized Correlation Function. *International Journal of Electronics and Telecommunication* 71, 3 (July 2025). https://ijet.pl/index.php/ijet/article/view/10.24425-ijet.2025.153627

[111] Elon Musk and Neuralink. 2019. An Integrated Brain-Machine Interface Platform With Thousands of Channels. *Journal of Medical Internet Research* 21, 10 (Oct. 2019). https://doi.org/10.2196/16194

[112] Arrate Muñoz-Barrutia, Raphaël Ertlé, and Michael Unser. 2002. Continuous Wavelet Transform with Arbitrary Scales and O(N) Complexity. *Signal Processing* 82 (May 2002). https://doi.org/10.1016/S0165-1684(02)00140-8

[113] Masaharu Nakashima. 1972. On the Propagation of Error in Numerical Integration. *Proceedings of the Japan Academy* 48, 7 (1972). https://doi.org/10.3792/pja/1195519594

[114] Noman Naseer and Keum-Shik Hong. 2015. fNIRS-based brain-computer interfaces: a review. *Frontiers in Human Neuroscience* 9 (Jan. 2015). https://doi.org/10.3389/fnhum.2015.00003

[115] Neuralink. 2024. PRIME Study Progress Update — User Experience. https://neuralink.com/blog/prime-study-progress-update-user-experience/

[116] Nicoletta Nicolaou and Julius Georgiou. 2010. Permutation Entropy: A new feature for Brain-Computer Interfaces. In *2010 Biomedical Circuits and Systems Conference (BioCAS)*. https://doi.org/10.1109/BIOCAS.2010.5709568

[117] Anna M. Nobili. 1988. The Accumulation of Integration Error. In *Long-Term Dynamical Behaviour of Natural and Artificial N-Body Systems*, Archie E. Roy (Ed.). Springer Netherlands, Dordrecht. https://doi.org/10.1007/978-94-009-3053-7_7

[118] Masako Omachi and Shinichiro Omachi. 2007. Fast calculation of continuous wavelet transform using polynomial. In *2007 International Conference on Wavelet Analysis and Pattern Recognition*, Vol. 4. https://doi.org/10.1109/ICWAPR.2007.4421725

[119] Antonio Ortega-Martinez, Alexander Von Lühmann, Parya Farzam, De'Ja Rogers, Emily M. Mugler, David A. Boas, and Meryem A. Yücel. 2022. Multivariate Kalman filter regression of confounding physiological signals for real-time classification of fNIRS data. *Neurophotonics* 9, 02 (June 2022). https://doi.org/10.1117/1.NPh.9.2.025003

[120] Dinh Q. Phung, Dat Tran, Wanli Ma, Phuoc Nguyen, and Tien Pham. 2014. Using Shannon Entropy as EEG Signal Feature for Fast Person Identification. https://openreview.net/forum?id=XMnKY8ItbE

[121] John Robinson Pierce. 1980. *An introduction to information theory: symbols, signals and noise* (2nd, rev. edition ed.). Dover publications, New York.

[122] Gaëlle Piret, Clément Hébert, Jean-Paul Mazellier, Lionel Rousseau, Emmanuel Scorsone, Myline Cottance, Gaelle Lissorgues, Marc O. Heuschkel, Serge Picaud, Philippe Bergonzo, and Blaise Yvert. 2015. 3D-nanostructured boron-doped diamond for microelectrode array neural interfacing. *Biomaterials* 53 (June 2015), 173–183. https://doi.org/10.1016/j.biomaterials.2015.02.021

[123] Tobias Pistohl, Andreas Schulze-Bonhage, Ad Aertsen, Carsten Mehring, and Tonio Ball. 2012. Decoding natural grasp types from human ECoG. *NeuroImage* 59, 1 (Jan. 2012). https://doi.org/10.1016/j.neuroimage.2011.06.084

[124] Adrien B Rapeaux and Timothy G Constandinou. 2021. Implantable brain machine interfaces: first-in-human studies, technology challenges and trends. *Current Opinion in Biotechnology* 72 (Dec. 2021). https://doi.org/10.1016/j.copbio.2021.10.001

[125] Benjamin I. Rapoport, Lorenzo Turicchia, Woradorn Wattanapanitch, Thomas J. Davidson, and Rahul Sarpeshkar. 2012. Efficient Universal Computing Architectures for Decoding Neural Activity. *PLoS ONE* 7, 9 (Sept. 2012). https://doi.org/10.1371/journal.pone.0042492

[126] Aravind Ravi, Nargess Heydari, and Ning Jiang. 2019. User-Independent SSVEP BCI Using Complex FFT Features and CNN Classification. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. https://doi.org/10.1109/SMC.2019.8914258

[127] O. Rioul and P. Duhamel. 1992. Fast algorithms for discrete and continuous wavelet transforms. *IEEE Transactions on Information Theory* 38, 2 (March 1992). https://doi.org/10.1109/18.119724

[128] Timothee Robert, MohammadAli Shaeri, and Mahsa Shoaran. 2025. BiND: A Neural Discriminator-Decoder for Accurate Bimanual Trajectory Prediction in Brain-Computer Interfaces. https://doi.org/10.48550/ARXIV.2509.03521

[129] Neethu Robinson, Cuntai Guan, and A P Vinod. 2015. Adaptive estimation of hand movement trajectory in an EEG based brain–computer interface system. *Journal of Neural Engineering* 12, 6 (Dec. 2015). https://doi.org/10.1088/1741-2560/12/6/066019

[130] Gennaro Rodrigues, Fernanda Lima Kastensmidt, Alberto Bosio, Gennaro Rodrigues, Fernanda Lima Kastensmidt, and Alberto Bosio. 2020. Survey on Approximate Computing and Its Intrinsic Fault Tolerance. *Electronics* 9, 4 (March 2020). https://doi.org/10.3390/electronics9040557

[131] Vincent Rouanne, Thomas Costecalde, Alim Louis Benabid, and Tetiana Aksenova. 2022. Unsupervised adaptation of an ECoG based brain–computer interface using neural correlates of task performance. *Scientific Reports* 12, 1 (Dec. 2022). https://doi.org/10.1038/s41598-022-25049-w

[132] José Rouillard, Alban Duprès, François Cabestaing, Stéphanie Leclercq, Marie-Hélène Bekaert, Charlotte Piau, Jean-Marc Vannobel, and Claudine Lecocq. 2015. Hybrid BCI Coupling EEG and EMG for Severe Motor Disabilities. *Procedia Manufacturing* 3 (Jan. 2015). https://doi.org/10.1016/j.promfg.2015.07.104

[133] Joe Saad. 2023. Architecture d'accélérateur matériel de calcul pour le décodage de signaux neuronaux dans une interface cerveau-machine. https://cea.hal.science/cea-04662630

[134] Joe Saad, Adrian Evans, Ilan Jaoui, Victor Roux-Sibillon, Emmanuel Hardy, and Lorena Anghel. 2025. Comparison metrics and power trade-offs for BCI motor decoding circuit design. *Frontiers in Human Neuroscience* 19 (March 2025). https://doi.org/10.3389/fnhum.2025.1547074

[135] Joe Saad, Adrian Evans, Ivan Miro-Panades, Tetiana Aksenova, and Lorena Anghel. 2025. Low-Power Feature Extraction for BCI Applications Using Approximate Continuous Wavelet Transform. In *2025 13th International Conference on Brain-Computer Interface (BCI)*. https://doi.org/10.1109/BCI65088.2025.10931272

[136] Joe Saad, Adrian Evans, Ivan Miro Panades, Tetiana Aksenova, and Lorena Anghel. 2023. Towards Low-Power Embedded ECoG Decoding. https://cea.hal.science/cea-04487786

[137] Joe Saad, Adrian Evans, Victor Roux-Sibillon, Ivan Miro-Panades, Tetiana Aksenova, and Lorena Anghel. 2025. An Adaptive Embedded Platform to Enable Real-Time Brain Motor Decoding. In *2025 21st International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*. https://doi.org/10.1109/DCOSS-IoT65416.2025.00019

[138] Joe Saad, Ivan Miro-Panades, Adrian Evans, and Lorena Anghel. 2026. ALIFE-BCI: An Adaptive Low-power Integrated Feature Extractor for Brain-Computer Interfaces. In *2026 Design, Automation & Test in Europe Conference (DATE)*. 1–7. https://doi.org/10.23919/DATE69613.2026.11539599 ISSN: 1558-1101.

[139] Siavash Sakhavi, Cuntai Guan, and Shuicheng Yan. 2015. Parallel convolutional-linear neural network for motor imagery classification. In *2015 23rd European Signal*

Processing Conference (EUSIPCO). https://doi.org/10.1109/EUSIPCO.2015.7362882

[140] Jacob B. Salyers, Yue Dong, and Yan Gai. 2019. Continuous Wavelet Transform for Decoding Finger Movements From Single-Channel EEG. *IEEE Transactions on Biomedical Engineering* 66, 6 (June 2019). https://doi.org/10.1109/TBME.2018.2876068

[141] Justin C. Sanchez, Paul R. Carney, and Jose C. Principe. 2006. Analysis of Amplitude Modulated Control Features for ECoG Neuroprosthetics. In *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*. https://doi.org/10.1109/IEMBS.2006.259587

[142] Marie-Caroline Schaeffer and Tetiana Aksenova. 2016. Switching Markov decoders for asynchronous trajectory reconstruction from ECoG signals in monkeys for BCI applications. *Journal of Physiology-Paris* 110, 4, Part A (Nov. 2016). https://doi.org/10.1016/j.jphysparis.2017.03.002

[143] G. Schalk, J. Kubánek, K. J. Miller, N. R. Anderson, E. C. Leuthardt, J. G. Ojemann, D. Limbrick, D. Moran, L. A. Gerhardt, and J. R. Wolpaw. 2007. Decoding two-dimensional movement trajectories using electrocorticographic signals in humans. *Journal of Neural Engineering* 4, 3 (June 2007). https://doi.org/10.1088/1741-2560/4/3/012

[144] MohammadAli Shaeri, Uisub Shin, Amitabh Yadav, Riccardo Caramellino, Gregor Rainer, and Mahsa Shoaran. 2024. A 2.46-mm ^2 Miniaturized Brain–Machine Interface (MiBMI) Enabling 31-Class Brain-to-Text Decoding. *IEEE Journal of Solid-State Circuits* (2024). https://doi.org/10.1109/JSSC.2024.3443254

[145] Niall Mc Shane, Karl McCreadie, Darryl Charles, Attila Korik, and Damien Coyle. 2023. Decoding Motion Trajectories in an Upper Limb BCI: Linear Regression vs Deep Learning. In *2023 IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE)*. https://doi.org/10.1109/MetroXRAINE58569.2023.10405752

[146] R. Shelishiyah, Deepa Beeta Thiyam, M. Jehosheba Margaret, and N. M. Masoodhu Banu. 2025. A hybrid CNN model for classification of motor tasks obtained from hybrid BCI system. *Scientific Reports* 15, 1 (Jan. 2025). https://doi.org/10.1038/s41598-024-84883-2

[147] M.J. Shensa. 1992. The discrete wavelet transform: wedding the a trous and Mallat algorithms. *IEEE Transactions on Signal Processing* 40, 10 (Oct. 1992). https://doi.org/10.1109/78.157290

[148] Duk Shin, Hidenori Watanabe, Hiroyuki Kambara, Atsushi Nambu, Tadashi Isa, Yukio Nishimura, and Yasuharu Koike. 2012. Prediction of Muscle Activities from Electrocorticograms in Primary Motor Cortex of Primates. *PLoS ONE* 7, 10 (Oct. 2012). https://doi.org/10.1371/journal.pone.0047992

[149] Uisub Shin, Cong Ding, Bingzhao Zhu, Yashwanth Vyza, Alix Trouillet, Emilie C. M. Revol, Stéphanie P. Lacour, and Mahsa Shoaran. 2022. NeuralTree: A 256-Channel 0.227-μJ/Class Versatile Neural Activity Classification and Closed-Loop

Neuromodulation SoC. *IEEE Journal of Solid-State Circuits* 57, 11 (Nov. 2022). https://doi.org/10.1109/JSSC.2022.3204508

[150] Mehdi Shokoueinejad, Dong-Wook Park, Yei Hwan Jung, Sarah Brodnick, Joseph Novello, Aaron Dingle, Kyle Swanson, Dong-Hyun Baek, Aaron Suminski, Wendell Lake, Zhenqiang Ma, and Justin Williams. 2019. Progress in the Field of Micro-Electrocorticography. *Micromachines* 10 (Jan. 2019). https://doi.org/10.3390/mi10010062

[151] Avishek Sinha Roy and Anindya Sundar Dhar. 2020. SIBAM—Sign Inclusive Broken Array Multiplier Design for Error Tolerant Applications. *IEEE Transactions on Circuits and Systems II: Express Briefs* 67, 11 (Nov. 2020). https://doi.org/10.1109/TCSII.2020.2969903

[152] Siuly, Hua Wang, and Yanchun Zhang. 2016. Detection of motor imagery EEG signals employing Naïve Bayes based learning process. *Measurement* 86 (May 2016). https://doi.org/10.1016/j.measurement.2016.02.059

[153] Surjo R. Soekadar, Matthias Witkowski, Nicola Vitiello, and Niels Birbaumer. 2015. An EEG/EOG-based hybrid brain-neural computer interaction (BNCI) system to control an exoskeleton for the paralyzed hand. *Biomedical Engineering / Biomedizinische Technik* 60, 3 (June 2015). https://doi.org/10.1515/bmt-2014-0126

[154] Marina Sokolova and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. *Information Processing & Management* 45, 4 (July 2009). https://doi.org/10.1016/j.ipm.2009.03.002

[155] M. Spüler, A. Walter, A. Ramos-Murguialday, G. Naros, N. Birbaumer, A. Gharabaghi, W. Rosenstiel, and M. Bogdan. 2014. Decoding of motor intentions from epidural ECoG recordings in severely paralyzed chronic stroke patients. *Journal of Neural Engineering* 11, 6 (Oct. 2014). https://doi.org/10.1088/1741-2560/11/6/066008

[156] Srinivasa R. Sridhara, Michael DiRenzo, Srinivas Lingam, Seok-Jun Lee, Raúl Blazquez, Jay Maxey, Samer Ghanem, Yu-Hung Lee, Rami Abdallah, Prashant Singh, and Manish Goel. 2011. Microwatt Embedded Processor Platform for Medical System-on-Chip Applications. *IEEE Journal of Solid-State Circuits* 46, 4 (April 2011). https://doi.org/10.1109/JSSC.2011.2108910

[157] Aaron Stillmaker, Zhibin Xiao, and Bevan Baas. [n.d.]. Toward More Accurate Scaling Estimates of CMOS Circuits from 180 nm to 22 nm. ([n.d.]).

[158] R. G. Stockwell. 2007. A basis for efficient representation of the S-transform. *Digital Signal Processing* 17, 1 (Jan. 2007). https://doi.org/10.1016/j.dsp.2006.04.006

[159] Sushree Satvatee Swain, Dipti Patra, and Yengkhom Omesh Singh. 2020. Automated detection of myocardial infarction in ECG using modified Stockwell transform and phase distribution pattern from time-frequency analysis. *Biocybernetics and Biomedical Engineering* 40, 3 (July 2020). https://doi.org/10.1016/j.bbe.2020.06.004

[160] Wing-kin Tam, Tong Wu, Qi Zhao, Edward Keefer, and Zhi Yang. 2019. Human motor decoding from neural signals: a review. *BMC Biomedical Engineering* 1 (Sept. 2019). https://doi.org/10.1186/s42490-019-0022-z

[161] Simone Tanzarella, Massimiliano Iacono, Elisa Donati, Dario Farina, and Chiara Bartolozzi. 2023. Neuromorphic Decoding of Spinal Motor Neuron Behaviour During Natural Hand Movements for a New Generation of Wearable Neural Interfaces. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 31 (2023). https://doi.org/10.1109/TNSRE.2023.3295658

[162] Shweta Thakur, Samriti Thakur, Aryan Rana, Pankaj Kumar, Kranti Kumar, and Chien-Ming Chen. 2025. Exploring the Evolution of Feature Extraction Methods in Brain–Computer Interfaces (BCIs): A Systematic Review of Research Progress and Future Trends. *WIREs Data Mining and Knowledge Discovery* 15, 3 (2025). https://doi.org/10.1002/widm.70040

[163] Miao Tian, Shurui Li, Ren Xu, Andrzej Cichocki, and Jing Jin. 2025. An Interpretable Regression Method for Upper Limb Motion Trajectories Detection with EEG Signals. *IEEE Transactions on Biomedical Engineering* (2025). https://doi.org/10.1109/TBME.2025.3557255

[164] Ramin Toosi, Mohammad Ali Akhaee, and Mohammad-Reza A. Dehaqani. 2021. An automatic spike sorting algorithm based on adaptive spike detection and a mixture of skew-t distributions. *Scientific Reports* 11, 1 (July 2021). https://doi.org/10.1038/s41598-021-93088-w

[165] Adriano B. L. Tort, Robert Komorowski, Howard Eichenbaum, and Nancy Kopell. 2010. Measuring Phase-Amplitude Coupling Between Neuronal Oscillations of Different Frequencies. *Journal of Neurophysiology* 104, 2 (Aug. 2010). https://doi.org/10.1152/jn.00106.2010

[166] Stefano Tortora, Stefano Ghidoni, Carmelo Chisari, Silvestro Micera, and Fiorenzo Artoni. 2020. Deep learning-based BCI for gait decoding from EEG with LSTM recurrent neural network. *Journal of Neural Engineering* 17, 4 (July 2020). https://doi.org/10.1088/1741-2552/ab9842

[167] M. Unser, A. Aldroubi, and M. Eden. 1993. B-spline signal processing. I. Theory. *IEEE Transactions on Signal Processing* 41, 2 (Feb. 1993). https://doi.org/10.1109/78.193220

[168] Giuseppe Varone, Wadii Boulila, Maha Driss, Saru Kumari, Muhammad Khurram Khan, Thippa Reddy Gadekallu, and Amir Hussain. 2024. Finger pinching and imagination classification: A fusion of CNN architectures for IoMT-enabled BCI applications. *Information Fusion* 101 (Jan. 2024). https://doi.org/10.1016/j.inffus.2023.102006

[169] Carmen Vidaurre, Nicole Krämer, Benjamin Blankertz, and Alois Schlögl. 2009. Time Domain Parameters as a feature for EEG-based Brain–Computer Interfaces. *Neural Networks* 22, 9 (Nov. 2009). https://doi.org/10.1016/j.neunet.2009.07.020

[170] Jiri Vrba and Stephen E. Robinson. 2001. Signal Processing in Magnetoencephalography. *Methods* 25, 2 (Oct. 2001). https://doi.org/10.1006/meth.2001.1238

[171] Gilbert G. Walter and Xiaoping Shen. 2000. *Wavelets and Other Orthogonal Systems, Second Edition*. CRC Press.

[172] Boyu Wang, Feng Wan, Peng Un Mak, Pui In Mak, and Mang I Vai. 2009. EEG signals classification for brain computer interfaces based on Gaussian process classifier. In *2009 7th International Conference on Information, Communications and Signal Processing (ICICS)*. https://doi.org/10.1109/ICICS.2009.5397570

[173] Fufeng Wang, Zihe Luo, Wei Lv, and XiaoLin Zhu. 2025. DTCNet: finger flexion decoding with three-dimensional ECoG data. *Frontiers in Computational Neuroscience* 19 (July 2025). https://doi.org/10.3389/fncom.2025.1627819

[174] Po T. Wang, Everardo Camacho, Ming Wang, Yongcheng Li, Susan J. Shaw, Michelle Armacost, Hui Gong, Daniel Kramer, Brian Lee, Richard A. Andersen, Charles Y. Liu, Payam Heydari, Zoran Nenadic, and An H. Do. 2019. A benchtop system to assess the feasibility of a fully independent and implantable brain-machine interface. *Journal of Neural Engineering* 16, 6 (Nov. 2019). https://doi.org/10.1088/1741-2552/ab4b0c

[175] Po T. Wang, Keulanna Gandasetiawan, Colin M. McCrimmon, Alireza Karimi-Bidhendi, Charles Y. Liu, Payam Heydari, Zoran Nenadic, and An H. Do. 2016. Feasibility of an ultra-low power digital signal processor platform as a basis for a fully implantable brain-computer interface system. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. https://doi.org/10.1109/EMBC.2016.7591725

[176] Wei Wang, Jennifer L. Collinger, Alan D. Degenhart, Elizabeth C. Tyler-Kabara, Andrew B. Schwartz, Daniel W. Moran, Douglas J. Weber, Brian Wodlinger, Ramana K. Vinjamuri, Robin C. Ashmore, John W. Kelly, and Michael L. Boninger. 2013. An electrocorticographic brain interface in an individual with tetraplegia. *PloS One* 8, 2 (2013). https://doi.org/10.1371/journal.pone.0055344

[177] Zhongpeng Wang, Beibei He, Yijie Zhou, Long Chen, Bin Gu, Shuang Liu, Minpeng Xu, Feng He, and Dong Ming. 2022. Incorporating EEG and EMG Patterns to Evaluate BCI-Based Long-Term Motor Training. *IEEE Transactions on Human-Machine Systems* 52, 4 (Aug. 2022). https://doi.org/10.1109/THMS.2022.3168425

[178] Yongzhong Wen, Yuejun Zhang, Liang Wen, Haojie Cao, Guangpeng Ai, Minghong Gu, Pengjun Wang, and Huiling Chen. 2022. A 65nm/0.448 mW EEG processor with parallel architecture SVM and lifting wavelet transform for high-performance and low-power epilepsy detection. *Computers in Biology and Medicine* 144 (May 2022). https://doi.org/10.1016/j.compbiomed.2022.105366

[179] Francis R. Willett, Daniel R. Young, Brian A. Murphy, William D. Memberg, Christine H. Blabe, Chethan Pandarinath, Sergey D. Stavisky, Paymon Rezaii, Jad Saab, Benjamin L. Walter, Jennifer A. Sweet, Jonathan P. Miller, Jaimie M. Henderson, Krishna V. Shenoy, John D. Simeral, Beata Jarosiewicz, Leigh R. Hochberg,

Robert F. Kirsch, and A. Bolu Ajiboye. 2019. Principled BCI Decoder Design and Parameter Selection Using a Feedback Control Model. *Scientific Reports* 9, 1 (June 2019). https://doi.org/10.1038/s41598-019-44166-7

[180] Minho Won, Hassan Albalawi, Xin Li, and Donald E. Thomas. 2014. Low-power hardware implementation of movement decoding for brain computer interface with reduced-resolution discrete cosine transform. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. https://doi.org/10.1109/EMBC.2014.6943916

[181] Dongrui Wu, Jung-Tai King, Chun-Hsiang Chuang, Chin-Teng Lin, and Tzyy-Ping Jung. 2018. Spatial Filtering for EEG-Based Regression Problems in Brain–Computer Interface (BCI). *IEEE Transactions on Fuzzy Systems* 26, 2 (April 2018). https://doi.org/10.1109/TFUZZ.2017.2688423

[182] Huaqiang Wu, Zhengwu Liu, Jie Mei, Jianshi Tang, Minpeng Xu, Bin Gao, Kun Wang, Sanchuan Ding, Qi Liu, Qi Qin, Weize Chen, Yue Xi, Yijun Li, Peng Yao, Han Zhao, Ngai Wong, He Qian, Bo Hong, Tzyy-Ping Jung, and Dong Ming. 2024. Memristor chip-enabled adaptive neuromorphic decoder for co-evolutional brain-computer interfaces. https://doi.org/10.21203/rs.3.rs-3966063/v1

[183] Yongcheng Wu, Guanghua Xu, Yifan Wu, Bo Wang, Nan Duan, Liang Zeng, Zezhen Han, and Sicong Zhang. 2021. An Asynchronous Detection Algorithm for SSVEP-Based BCI Using Gradient Boosting Decision Tree. In *Proceedings of the 2020 9th International Conference on Computing and Pattern Recognition (ICCPR '20)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3436369.3437405

[184] Cheng-Yi Xiong, Jin-Wen Tian, and Jian Liu. 2006. Efficient high-speed/low-power line-based architecture for two-dimensional discrete wavelet transform using lifting scheme. *IEEE Transactions on Circuits and Systems for Video Technology* 16, 2 (Feb. 2006). https://doi.org/10.1109/TCSVT.2005.860121

[185] Fang Zhou Xu, Wen Feng Zheng, Dong Ri Shan, Qi Yuan, and Wei Dong Zhou. 2020. Decoding spectro-temporal representation for motor imagery recognition using ECoG-based brain-computer interfaces. *Journal of integrative neuroscience* 19, 2 (June 2020). https://doi.org/10.31083/j.jin.2020.02.1269

[186] Jerald Yoo, Long Yan, Dina El-Damak, Muhammad Awais Bin Altaf, Ali H. Shoeb, and Anantha P. Chandrakasan. 2013. An 8-Channel Scalable EEG Acquisition SoC With Patient-Specific Seizure Classification and Recording Processor. *IEEE Journal of Solid-State Circuits* 48, 1 (Jan. 2013). https://doi.org/10.1109/JSSC.2012.2221220

[187] Do-Yeon Yoon, Sonal Pinto, SungWon Chung, Paul Merolla, Thong-Wei Koh, and Dongjin Seo. 2021. A 1024-Channel Simultaneous Recording Neural SoC with Stimulation and Real-Time Spike Detection. In *2021 Symposium on VLSI Circuits*. https://doi.org/10.23919/VLSICircuits52068.2021.9492480

[188] Mohamad Amin Younessi Heravi, Keivan Maghooli, Fereidoun Nowshiravan Rahatabad, and Ramin Rezaee. 2023. A New Nonlinear Autoregressive Exogenous

(NARX)-based Intra-spinal Stimulation Approach to Decode Brain Electrical Activity for Restoration of Leg Movement in Spinally-injured Rabbits. *Basic and Clinical Neuroscience* 14, 1 (Jan. 2023). https://doi.org/10.32598/bcn.2022.1840.1

[189] Fan Zhang, Apurva Mishra, Andrew G. Richardson, and Brian Otis. 2011. A Low-Power ECoG/EEG Processing IC With Integrated Multiband Energy Extractor. *IEEE Transactions on Circuits and Systems I: Regular Papers* 58, 9 (Sept. 2011). https://doi.org/10.1109/TCSI.2011.2163972

[190] Yue Zhang, Sheng Quan Xie, He Wang, Chaoyang Shi, and Zhi-Qiang Zhang. 2023. Bayesian-Based Classification Confidence Estimation for Enhancing SSVEP Detection. *IEEE Transactions on Instrumentation and Measurement* 72 (2023). https://doi.org/10.1109/TIM.2023.3284952

[191] Zheng Zhang and Timothy G. Constandinou. 2023. Firing-rate-modulated spike detection and neural decoding co-design. *Journal of Neural Engineering* 20, 3 (May 2023). https://doi.org/10.1088/1741-2552/accece

[192] Hai-bin Zhao, Chun-yang Yu, Chong Liu, and Hong Wang. 2010. ECoG-based brain-computer interface using relative wavelet energy and probabilistic neural network. In *2010 3rd International Conference on Biomedical Engineering and Informatics*, Vol. 2. https://doi.org/10.1109/BMEI.2010.5639897

[193] Zhiwei Zhong, Yijie Wei, Lance Christopher Go, and Jie Gu. 2024. 33.2 A Sub-1μJ/class Headset-Integrated Mind Imagery and Control SoC for VR/MR Applications with Teacher-Student CNN and General-Purpose Instruction Set Architecture. In *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 67. https://doi.org/10.1109/ISSCC49657.2024.10454317

Appendices

A Background on Clinatec BCI Research

This appendix provides complementary context on the Brain-Computer Interface (BCI) research program at Clinatec, within which the work of this thesis was done. It highlights the clinical motivation, the hardware-software platform, and the explored Phase-Amplitude Coupling (PAC) research topic.

A.1 Clinical Motivation

The Clinatec program aims to restore lost motor function for patients suffering from a spinal cord injury, through an Electrocorticography (ECoG)-based BCI combined with spinal cord stimulation. Electrode arrays are surgically placed on the surface of the brain to capture cortical activity, which is processed by an external compute unit to decode motor intentions. The decoded commands are then used to control effectors such as spinal stimulators or exoskeletons. Early clinical trials demonstrated that such closed-loop systems can enable patients to regain control of upper- and lower-limb movements [20, 88], showing feasibility in human subjects.

However, making such systems available for home use requires them to be portable and energy-efficient, while being capable of real-time operation. Furthermore, as the brain-machine connection evolves over time, the solution needs to adapt in real-time to preserve a high decoding accuracy, and learn new mental tasks.

A.2 BCI Platform

An initial research prototype consisted of a notebook computer, running Matlab R2022b and consuming $\approx$ 50W, to perform the signal decoding and update algorithms. This platform is not suitable for home use due to its large size and high power consumption. The experimental platform integrates three main components:

- A cortical ECoG implant for neural signal acquisition.
- A battery-powered compute unit responsible for decoding and user-interface tasks.
- A spinal stimulator that delivers stimulation patterns to restore movement.

Signals are transmitted wirelessly between implants and the compute unit. The compute unit performs the decoding and is also capable of updating the decoding model to adapt to neural plasticity or new tasks. These updates must be performed, along with the decoding, while respecting real-time constraints. More specifically, the decoding must be executed in less than 100 msec and the model updates in less than 15 seconds [88].

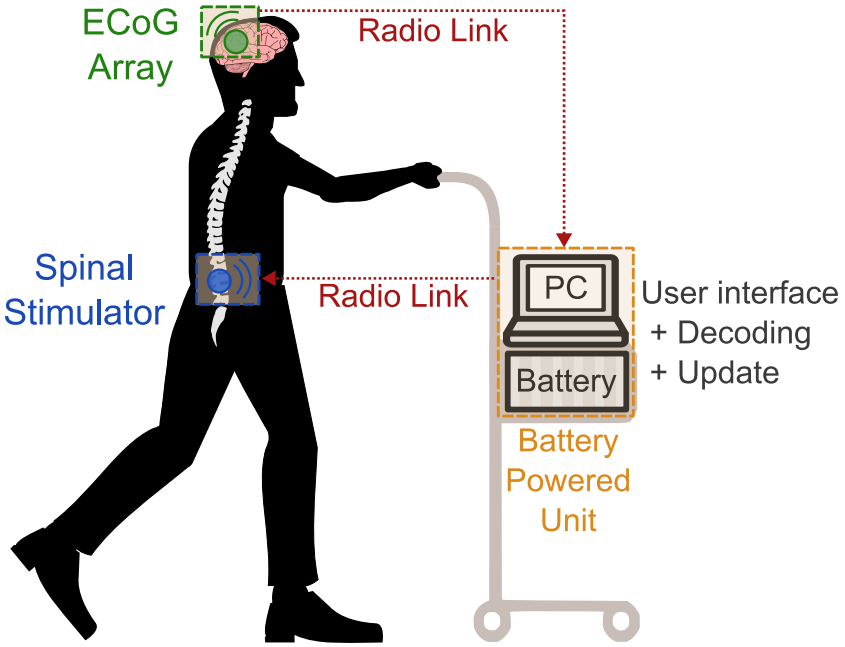


Figure A.1: Restoring Mobility After a Spinal Cord Injury Using an ECoG-Based Battery-Powered BCI and a Spinal Stimulator.

A.3 Phase-Amplitude Coupling Study

Beyond the time-frequency features used in the main BCI pipeline, we also investigated PAC as a potential source of information for motor decoding in long-term ECoG recordings [95]. PAC quantifies how the phase of a low-frequency oscillation modulates the amplitude of a higher-frequency component, providing a way to capture cross-frequency interactions that are not visible in amplitude alone [165].

The study used data from a clinical trial, in which a tetraplegic participant performed mental motor tasks to control a virtual avatar over multiple sessions [20]. Signals from 64 electrodes were decomposed in the time-frequency domain using a complex Continuous Wavelet Transform (CWT), and PAC was quantified with two coupling measures: the Modulation Index (MI) and the Mean Vector Length (MVL). First, PAC features were computed offline on windows of different lengths and used to classify pairs of states (idle vs right hand, idle vs left hand, right vs left hand) with linear classifiers. In a second stage, PAC features were computed using a sliding window and fed to a Markov Switching Linear Model (MSLM) decoder in a pseudo-online setup, either alone, or combined with standard amplitude features.

The results showed that PAC features can decode motor attempts with high accuracy in offline classification, especially when using MVL and longer windows [95]. The most informative couplings were theta/low-gamma and beta/high-gamma, with distinct spectral and spatial patterns. Regression analyses indicated that PAC shares some variance with gamma-band amplitude, but also carries complementary information. In the pseudo-online setting, PAC-based decoding remained efficient but did not outperform amplitude-based decoding, and combining both feature types did not significantly increase accuracy, likely due to the limited window length.

In this collaboration, I contributed to defining the analysis method, implementing Matlab scripts for PAC computation in the existing pipeline. I also participated in writing and revising the manuscript [95]. Although PAC features are not used in the main low-

power BCI architecture proposed in this thesis, this study showed that cross-frequency features can improve motor decoding, which motivates efficient implementations capable of handling complex feature sets in future BCI systems.

B Porting of Matlab Algorithms to C++

This appendix documents the algorithmic work required to port the Clinatec Brain-Computer Interface (BCI) pipeline from Matlab to a C++ implementation suitable for embedded use. We describe the decoding algorithm used in closed-loop experiments, the operating scenarios considered during design, and the algorithmic optimizations that enabled real-time execution on embedded platforms while preserving decoding accuracy. These optimizations have been published in the 21st International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT 2025) [137] and in the Design, Automation and Test in Europe Conference [45].

B.1 Decoding Algorithm

We describe the decoding algorithm used in the BCI platform introduced in Appendix A.2. In this system, the decoding algorithm is executed every 100 msec and its flow [106] is shown in Figure B.1. The raw input signals are collected from $M = 64$ electrodes (32 from an implant on the left hemisphere and 32 from the right hemisphere), each providing data at a sampling rate of 590 samples per second.

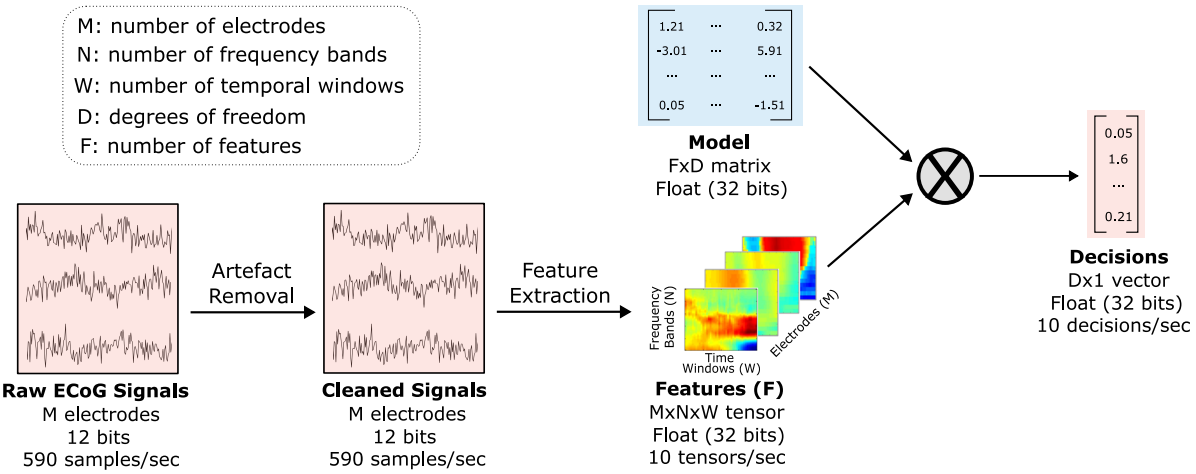


Figure B.1: ECoG Decoding Flow Showing the Steps from Signal Acquisition to Output Decisions

The first step in the process is to remove artefacts, which might result from errors caused by the Analog-to-Digital Converter (ADC) or the communication link, using filtering and thresholding. Then, a set of F features is extracted using a time-frequency analysis based on a Continuous Wavelet Transform (CWT). The input signal is convolved with a wavelet. The Morlet Wavelet is selected as it provides a good time-frequency resolution to identify specific motor intentions, and it is frequently used in BCI [87, 140]. The wavelet is scaled to cover N frequency bands in the range of 2 Hz to 200 Hz, and is then convolved with 100-millisecond windows of input data across M electrodes. The obtained time-frequency components are concatenated over W time windows, and the resulting features ($F = M \times N \times W$) are passed through a pre-calculated linear model mapping them to D Degrees of Freedom (DoFs). These DoFs can correspond to different movements of the arms or legs. A softmax function is applied to the output to compute the probability distribution of the possible classes, and a Hidden Markov Model (HMM)

ensures the stability of transitions between states. For continuous task decoding, the classification layer selects a linear model, "an expert", that estimates the continuous variable. In this paper, we mainly focus on the feature extraction and the linear model evaluation, which are the most compute intensive tasks.

B.1.1 Use-Case Scenarios

The system can be used for different applications including walking and 8-DoF continuous upper-limb movement decoding, which can be used to control an exoskeleton or a virtual avatar. We target both gait decoding and 8-DoF upper-limb control. In Table B.1 we list several scenarios, each requiring a different number of frequency bands and time windows, resulting in a different total number of features.

Table B.1: Characteristics of Use Scenarios

Scenario	**Application**	M (Electrodes)	N (Freq. bands)	W (Windows)
S1920	Walking	64	15	2
S3072	Walking	64	24	2
S7680	Walking	64	24	5
S9600	Upper limb (8-DoF)	64	15	10
S10752	Future use	64	24	7

The decoding accuracy has been previously evaluated for the different scenarios, using the Matlab R2022b code running on a notebook computer. For gait (walking) decoding (*S3072* and *S7680*), the model was able to achieve an accuracy of 74% when controlling six-joint flexion and extension movements (chance level is 14% for seven classes), and up to 97% for a simpler 3-state walking movement [88]. The same system, used to decode 8-DoF upper limb movements (*S9600*), achieved a success rate of 71% for an exoskeleton in lab conditions and 64% for a virtual avatar at home [20]. As this level of accuracy is sufficient for this BCI application, we mainly focus on porting the system to an embedded platform while preserving the same algorithmic approach and achieving the same accuracy. A first step to improve performance was to develop a bit-accurate C++ model to replace the Matlab R2022b version, and then further optimizations were applied to improve the run time of this code.

B.1.2 Optimizations for Feature Extraction

The decoding algorithm relies on time-frequency features computed with the Morlet-based CWT, already introduced in Chapter 3.1. Here, we focus on implementation details and optimizations for the embedded port. In practice, the Electrocorticography (ECoG) signals are sampled at 590 Hz. A 100 ms window therefore contains 59 samples; to improve the detection of low-frequency components and mitigate boundary effects, the most recent window is concatenated with the two previous ones, yielding an effective buffer of 177 (3×59) samples.

Feature extraction is performed using the Fast Fourier Transform (FFT)-based convolution method described in Section 3.1.1.3:

1. A FFT is performed on the input signal.
2. The signal FFT is multiplied by the pre-computed FFT of the wavelets.
3. An Inverse Fast Fourier Transform (iFFT) is performed to obtain the complex convolution result.
4. Energy is extracted by applying a $L2$-Norm on the complex outputs.

An initial benchmark of the decoding algorithm running in C++ on an x86 host for scenario *S7680* is shown in Fig. B.2 (see column 'No Optimization').

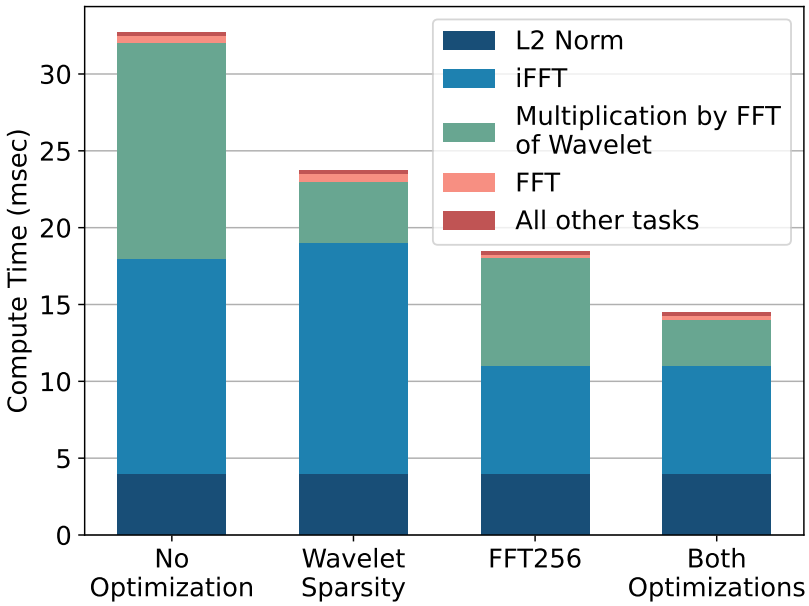


Figure B.2: Compute Time for Decoding in Scenario S7680, Running on an x86 Host.

Sparse Wavelet Multiplication

The FFT of the input data results in a complex, frequency-domain signal. Each complex multiplication between a wavelet coefficient and this signal requires four multiplications and two additions. The initial benchmark showed that these multiplications took close to half of the overall CPU time. In addition, the wavelet is, by definition, localized in both the time and frequency domains. Looking at a heat map of the wavelet FFT coefficients (Fig. B.3), we observe that the majority of the values are close, or extremely close, to zero for both the real and imaginary components. Skipping the multiplication of these near-zero values can improve the execution time without sacrificing decoding performance. Using a threshold of 10^{-11}, about 80% of the multiplications were skipped, resulting in a 25% reduction in the total execution time (see column 'Wavelet Sparsity' in Fig. B.2). This change barely impacted the decoding accuracy, resulting in a single misprediction in 2000 predictions, an error rate acceptable for the target applications.

FFT Size Reduction

A convolution between two signals requires a FFT length larger than the sum of their lengths. As FFT algorithms are more easily optimized for power-of-two lengths, the initial system required a 512-point FFT. However, the features are computed by using the overlapping part between the signal and the wavelet, which only keeps 59 points of the iFFT. The same points can be obtained by performing a 256-point FFT, for a signal of 177 points. Reducing the FFT size from 512 to 256 yielded close to a 50%

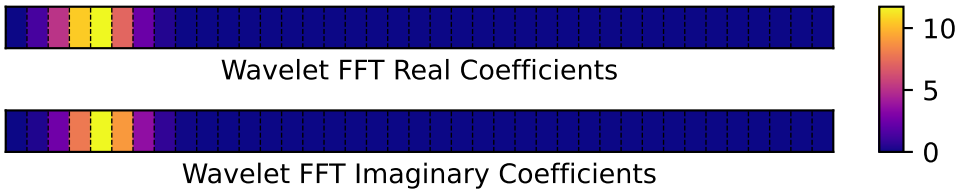


Figure B.3: Amplitude of the Real and Imaginary Parts of Morlet Wavelet FFT Coefficients.

reduction in compute time for this step (see column 'FFT256' in Fig. B.2). Combined, both optimizations resulted in a reduction of the total execution time by 50%.

B.2 Model Update Algorithm

The model used in the BCI decoding is initially trained in a clinical setting. During this phase, the patient is requested, by a physiotherapist, to imagine performing specific movements, while the model is updated periodically, typically every 15 s, to quickly build the initial decoding model [43]. During each period, the decoding (inference) continues, while the model is updated. Note, the overall model update process cannot be pipelined because the output of the previous update is required to start the current update.

Although ECoG arrays are stable over long periods of time [77], the signal patterns for a given movement evolve over time, the result of brain plasticity. In addition, by adapting to the BCI, the patient can gradually learn new mental tasks. As a result, the model must be periodically (after several days) updated to effectively decode the intention for each movement. For the patient, it is desirable that the periodic updates be performed in a home setting, requiring the embedded platform to have the compute capacity to execute the model update algorithm.

B.2.1 Description

The high-level flow of the update algorithm is shown in Fig. B.4. Every 100 ms, F features are computed, so after 15 s, a feature matrix X_0, of size $F \times 150$ is generated. Associated with X_0 is Y_0, the label matrix corresponding to the requested movements.

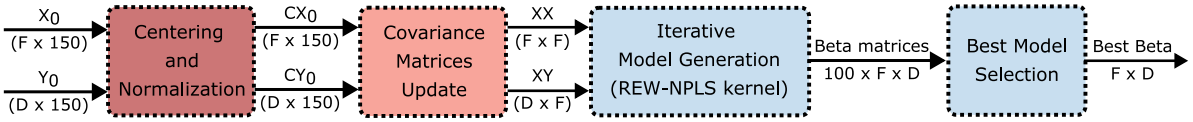


Figure B.4: High-Level Flow of the REW-NPLS Model Update Algorithm (15 s Update).

The first step of the algorithm consists of centering and normalizing both input matrices (subtracting the mean and dividing by the variance) over the time dimension, to obtain CX_0 and CY_0. The next step consists of computing the covariance matrices, as shown in Equations (B.1) and (B.2). The variable α is called the *forgetting factor* and controls the weighting between the old and the new covariance matrices.

$$XX_{new} = \alpha \cdot XX_{old} + CX_0 \cdot CX_0^T \tag{B.1}$$

$$XY_{new} = \alpha \cdot XY_{old} + CX_0 \cdot CY_0^T \tag{B.2}$$

The third step, the kernel of the algorithm, consists of iteratively building a projection of the data into a latent space. This requires obtaining the eigenvectors of the matrix $XY^T \cdot XY$ and applying the PARAFAC decomposition algorithm [26]. An empirical limit of 100 iterations has been set, as a latent space of dimension 100 is sufficient for brain signal decoding [43]. The obtained models are subject to overfitting. To avoid this, as a last step, each of the 100 linear models obtained during the iterations is tested on new data, and the best model is selected.

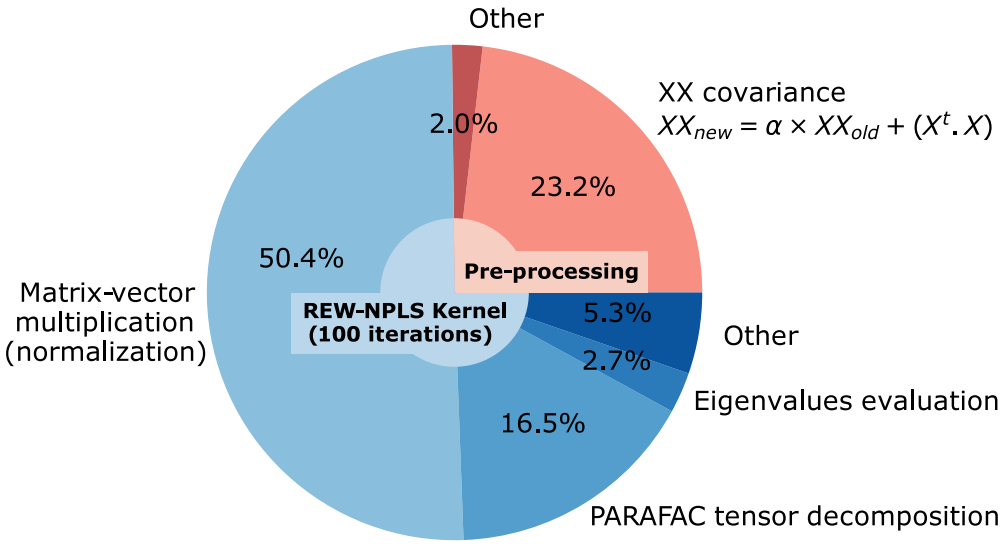


Figure B.5: Distribution of Compute Time for the Model Update Algorithm.

The initial compute time breakdown for implementing the model update algorithm is shown in Fig. B.5. The portion in red (25.2%) corresponds to the computation of the covariance matrices, and the larger part in blue (74.8%) corresponds to the kernel of the REW-NPLS algorithm. The latter is dominant and within this portion, one matrix operation, corresponding to the XX normalization, is the most time consuming.

B.2.2 Optimizations

We coded an initial version of the algorithm in C++, using the Armadillo library [1] for matrix operations, and compiled it to run on an embedded platform with an ARM A72 core. This initial version was not able to meet the real-time requirement of updating the model every 15 s, as seen on the bar labelled 'initial' in Fig. B.6.

Memory Optimization

Since the key operation in this algorithm is dense matrix multiplication, we reviewed how the matrices are stored in memory, to reduce the penalty due to cache misses. The Armadillo library stores matrices in *column-first* format. When performing matrix multiplication, one matrix is traversed by rows and the other by columns. A row traversal of a large matrix stored in *column-first* format is inefficient. To read each entry, a full cache line (64B) is read, but only one value in the line is actually used. It is preferable to work with the transpose of one of the matrices, so that both are traversed in the direction of the columns. Based on this insight, the code was modified to use the matrix transpose, when

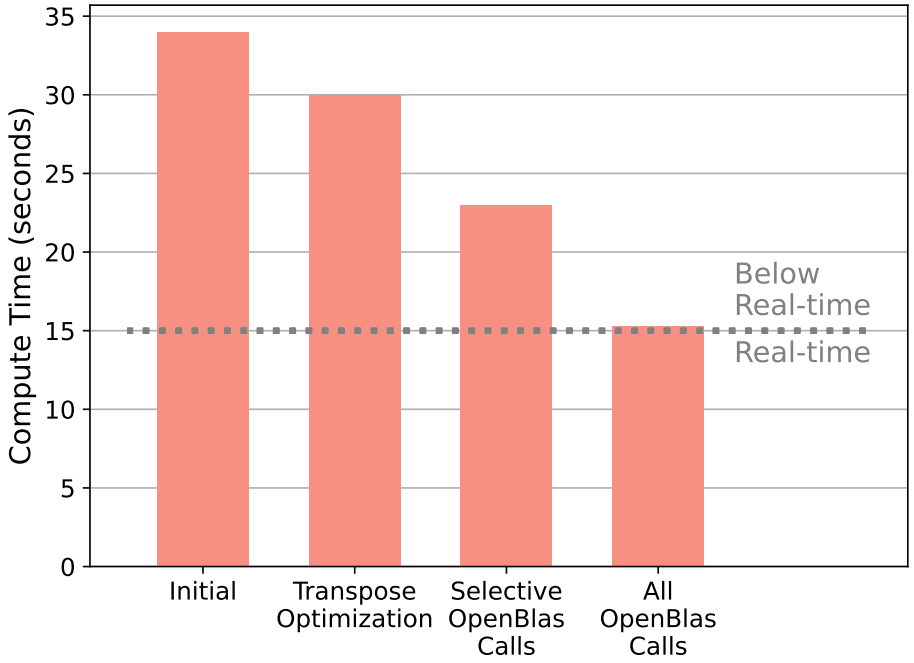


Figure B.6: Acceleration of the Update Algorithm (Scenario S7680, A72 Platform).

possible, resulting in a reduction of the update time from 34 to 30 s; a first improvement, but still far from the target time of 15 s.

Operation Optimization

OpenBLAS [3] is a library that implements basic linear algebra operations using highly optimized code that fully exploits the specific hardware features of the target processor, such as Single Instruction Multiple Data (SIMD) instructions. The computation of the covariance matrix shown in Equation B.1 can be directly implemented with the OpenBLAS function `dgemm` that performs $C \leftarrow \alpha \cdot A \cdot B + \beta \cdot C$. The code was modified to selectively make direct calls to the OpenBLAS library for the most critical operations, which resulted in a reduction of the update compute time to 23 s. Finally, these calls to OpenBLAS were used for all matrix operations, resulting in an update time near 15 s on the A72 platform, meaning the real-time target had been achieved for scenario S7680.

B.3 Experimental Measurements

B.3.1 Embedded Platforms

Two embedded platforms were evaluated as candidates for a future portable medical device. One platform is based on a BCM2711 System-on-Chip (SoC) and the other on the BCM2712C1 SoC. The characteristics of both are listed in Table B.2.

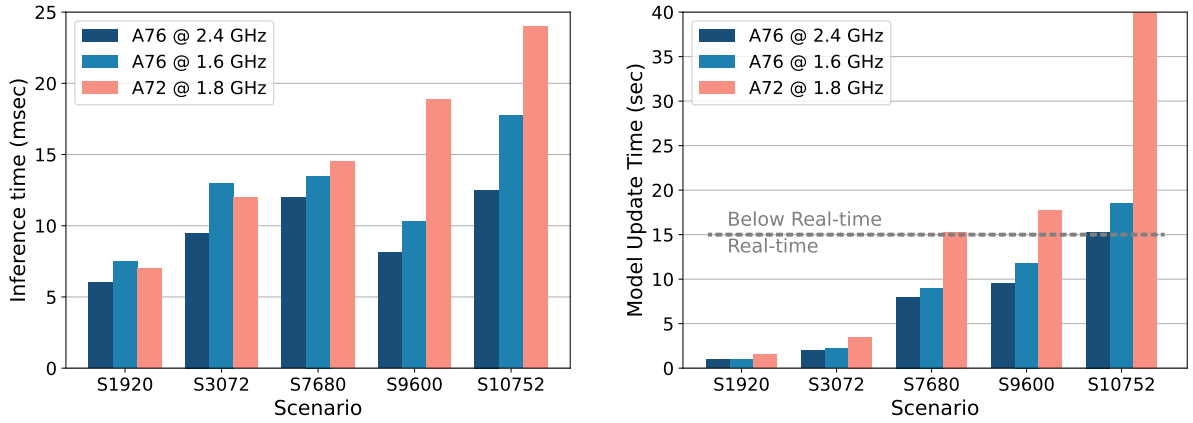
Both SoCs are based on quad-core ARM™ processors. The same SoC is also responsible for running the user interface and managing the radio links. To ensure real-time constraints are respected and to maintain task independence, one core is allocated to the decoding task and another to the model update process, leaving two cores free for other tasks.

Table B.2: Compute Capacity of Two Embedded Platforms.

	A72 system	**A76 system**
SoC reference	BCM2711	BCM2712C1
Processor	Quad Cortex™ A72	Quad Cortex™ A76
Core peak frequency	1.8 GHz	2.4 GHz
L1 data cache	32 KB	64 KB
L1 instruction cache	48 KB	64 KB
L2 cache	1 MB	512 KB / core
L3 cache	-	2 MB
DRAM capacity	8 GB	8 GB
DRAM speed	LPDDR4-2400	LPDDR4X

B.3.2 Execution Time

The optimized algorithms were tested on both platforms across all scenarios. The A76 platform was run at two core frequencies (1.6 GHz and 2.4 GHz).



(a) Inference Time across Five Scenarios. (b) Model Update Time across Five Scenarios.

Figure B.7: Execution Time on Two Embedded Platforms (A72 and A76) for (a) Inference and (b) Model Update.

Figure B.7a shows the execution time of the inference algorithm. The most compute-intensive tasks during inference (FFT, iFFT) scale with the number of frequencies. Thus, scenarios *S1920* and *S9600* are faster on the A76 platform, as they require only 15 frequency bands, compared to 24 bands for the others. The A72 platform appears to be limited by memory bandwidth, as the runtime grows directly with the number of features. Both systems are nevertheless able to easily achieve the 100 ms target for inference.

Figure B.7b shows the execution time of the model update algorithm on both platforms. For the first three scenarios, the A72 SoC maintains the 15 s real-time update constraint. For the most demanding scenarios, only the A76 SoC can meet the performance target. Profiling with `perf` (Table B.3) shows that the limiting factor on the A72

platform is memory bandwidth and cache size. Indeed, the A76 at 1.6 GHz performs the update faster than the A72 at 1.8 GHz. The larger caches of the A76 reduce page faults, and the L1 bandwidth is approximately twice that of the A72.

Table B.3: Comparison of Cortex A72 and A76 SoCs for Scenario *S7680*.

	A72 (1.8 GHz)	**A76 (2.4 GHz)**	**A76 (1.6 GHz)**
Update time	14.55 s	7.285 s	8.892 s
Page faults (N)	163,851	41,122	41,124
CPU frequency	1.777 GHz	2.359 GHz	1.575 GHz
Instructions/cycle	0.61	1.29	1.43
L1 transfer rate (Mb/s)	914	1882	1402
L1 miss rate	0.54%	2.11%	1.60%

B.3.3 Energy Measurements

When operating, the system is continuously performing inference, and when a model update is required (at time zero), the power increases while executing the extra load associated with the model update. The power profile over a 15 s time window is shown in Fig. B.8, where both systems are processing scenario *S7680*.

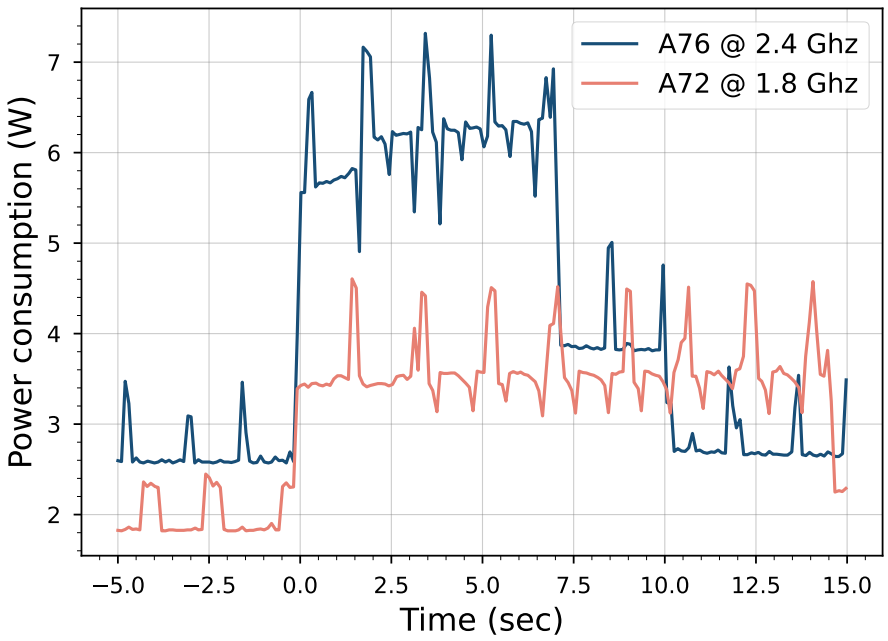


Figure B.8: Power Consumption vs. Time for Both Platforms.

It can be seen that the A72 system consumes 1.9 W for inference-only mode. However, when the model update starts, the power consumption increases to 3.5 W during the whole update cycle, as the process takes the full 15 s. For the A76 system, the average power during inference is higher (2.7 W). During update, the power consumption (6.2 W) is

much higher than that of the A72, however, it takes less time (7.5 s). When the model update is completed, the core running the update algorithm returns to the idle state, while the one running inference keeps running. Thus, the power first drops to 3.9 W and then goes back to 2.7 W. We believe this extra step is an artifact of the Dynamic Voltage/Frequency Scaling (DVFS) hardware in the A76 SoC.

Table B.4 shows the mean power consumption for continuous inference and continuous model update. The power was integrated over time to compute the energy required over a 15 s window. Although the A76 SoC performs the update much faster than the A72 SoC, over the 15 s window, it requires 28% more energy. If the system is powered by a standard 26,800 mAh battery, the A72 (resp. A76) platform would achieve 27 h (resp. 21 h) of continuous model update.

Table B.4: Power and Energy Comparison between A72 and A76 SoCs for Scenario *S7680*.

	A72 (1.8 GHz)	**A76 (2.4 GHz)**
Mean power inference	1.9 W	2.7 W
Mean power update (15 s)	3.5 W	4.7 W
Energy update (15 s)	53 J	71 J
Battery life (continuous update)	27 h	21 h

For some application scenarios (Fig. B.7b), the A72 SoC meets the real-time requirement for model updates and increases the battery life. Moreover, the A76 platform meets the requirements for all tested scenarios at the expense of a reduced battery lifetime. The higher performance of this platform allows it to address more complex scenarios. Compared to the initial laptop, both proposed platforms consume ≈10× less power while performing model updates.

C Complex-To-Real (C2R) Post-processing Block

The purpose of the Complex-to-Real (C2R) block is to convert complex-valued data produced by the Continuous Wavelet Transform (CWT) into real-valued features, optionally in floating-point precision, suitable for decoding models. In the Clinatec algorithm, this conversion uses a L_2 norm shown in Figure C.1, although other scenarios may use L_1 or L_∞ norms.

$$y = \sqrt{x_{\text{re}}^2 + x_{\text{im}}^2}, \tag{C.1}$$

For debug purposes, it may also be useful to perform operations on the raw real or imaginary components.

C.1 Use Case and Integration

The C2R post-processing module was integrated into a full decoding circuit within the NEMO-BMI European project, where it operates after the convolution stage and before applying the decoder. Separately, Chapter 5 studies a different application that integrates the Adaptive Quality Feature Extractor (AQFE) into a closed-loop Brain-Computer Interface (BCI) system. The *Output RAM* is the interface between the C2R and the AQFE, as detailed in Section 5.1.

C.2 Datapath

The AQFE writes the complex convolution outputs into the *Output RAM*. The C2R module then reads those values, converts them into real-valued energy features with selectable norms, and forwards them to the downstream decoder. Processing starts when a command is received from the external interface. This command indicates a starting address and stride in the AQFE *Output RAM*, the number of words to process, the set of operations to be performed by the C2R, and the scaling offsets for those operations, as shown in the datapath in Figure C.1. In the current implementation, the square-root and integer-to-float opretions are implemented using Synopsys DesignWare™ modules.

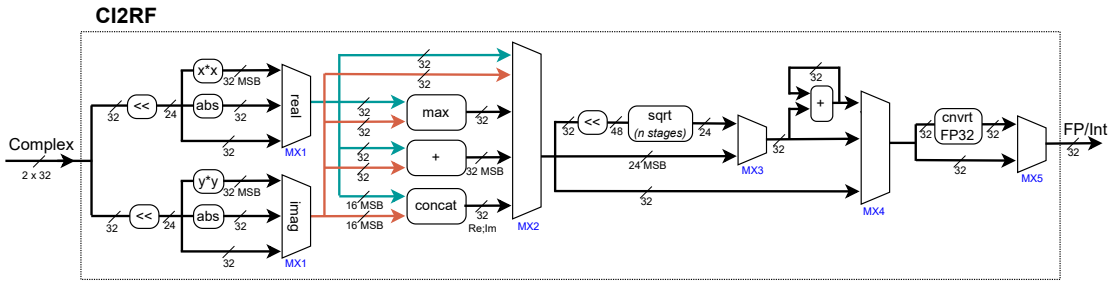


Figure C.1: Datapath of the C2R Module with Selectable Norms and Optional Scaling.

After receiving an external command, an address generation unit issues read requests to the *Output RAM*. As data is received, it is processed by a fixed-latency pipeline whose depth depends on the enabled operations. The block supports aggregation over custom time windows and an optional normalization step. In addition, by selecting proper configurations of the muxes, it is possible to compute L_1, L_2 or L_∞ norms.

C.3 Programming Commands

Table C.1 lists the fields that can be selected by the external interface. These fields contain the information required for read address generation, and for selecting the datapath operations, including window aggregation and optional scaling.

Table C.1: C2R External Interface Parameters

Field	Description
Num Words	Number of words to process (read and perform operations on).
Read Address	Base read address in AQFE *Output RAM* (complex convolution outputs).
Read Stride	Stride applied when reading from the source memory.
MX1	Real/Imag pre-operation: no-op, absolute value, or squaring.
MX2	Combine stage: concatenation, sum, maximum, or imaginary or real part.
MX3	Square-root stage: bypass or apply the square-root operation.
MX4	Accumulation stage: bypass from previous stage, full bypass, or accumulate.
MX5	Output format: bypass as fixed-point or convert to FP32.
Input Shift	Optional left shift on inputs before MX1.
Sqrt Shift	Optional left shift before the square root.
FP Offset	Signed offset added to the FP32 exponent (power-of-two scaling).

C.4 Throughput and Latency

At each clock cycle, a complex output is read by the address generation unit. The pipeline has a configuration-dependent latency, as all operators perform computation with a fixed number of clock cycles. Once filled, the pipeline sustains one output per cycle in the non-accumulation configuration.When accumulation is enabled, it produces a single output per processed window.

C.5 Silicon Area and Power Consumption

The C2R block was synthesized in the same 22 nm FDSOI technology used for the AQFE. The implementation targets a 100 MHz clock at 0.8 V and 25°C. We explicitly used DesignWare^{TM} square and square-root units in the RTL to optimize the most costly operations. Under these conditions, the silicon area of the C2R is approximately 0.0041 mm^2. Power analysis was performed using real BCI data as switching activity, in a configuration matching the target use case for the paradigm described in [20]. The C2R block processes 960 features at a 10 Hz decision rate, using the L_2 norm and a 32-bit floating-point output format, which corresponds to the most costly configuration. In this setting, the average dynamic power during computation is 7.34 μW, and the leakage power is 23.8 nW. Compared to the feature-extraction cost, the C2R post-processing occupies a small silicon area and adds a limited energy overhead while providing flexibility on the norm and output format.

D Approximate Multipliers

This appendix describes an exploratory path that was considered during the circuit design described in Chapter 4 but not pursued in the final design. The goal was to assess whether approximate multipliers could further reduce the energy cost of feature extraction, on top of the algorithmic savings obtained with approximate Continuous Wavelet Transform (CWT) methods.

D.1 Background on Approximate Multipliers

Many signal-processing and machine-learning applications do not require all arithmetic operations to be computed exactly. In these cases, arithmetic operators can be replaced by approximate compute modules to reduce area or power. Multipliers are often targeted, as they are among the most expensive operators in the datapath.

D.1.1 Approximation Methods

Approximation is typically introduced at the level of the partial-product generation and accumulation. Common strategies include truncating least-significant partial products or simplifying the adders for reduction. For example, truncated multipliers remove some of the least-significant columns of the partial products and can partially compensate the introduced error with corrections [75]. Other designs replace exact adders in the reduction tree by approximate versions with fewer logic gates [44]. Comparative studies such as [66] review different approximate multiplier architectures and evaluate them in terms of circuit characteristics. Several libraries and design methods have been proposed for approximate multipliers. The EvoApprox8b library [108] provides a large set of approximate adders and multipliers, together with error metrics for evaluation.

D.1.2 Broken-Array Multipliers

Broken-Array Multipliers (BAMs) approximate the multiplication by omitting some parts of the partial products. The array is divided by two cut levels, horizontal and vertical, which define truncated regions [151]. Bits above the horizontal cut or to the right of the vertical cut are omitted, reducing the number of adders and the switching activity required to evaluate the product. Figure D.1, from [65], illustrates this principle, with an example of vertical and horizontal cuts.

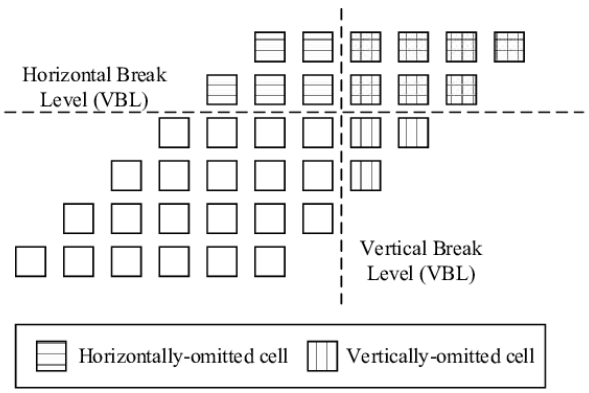


Figure D.1: Example of a BAM with Horizontal and Vertical Cuts, Adapted from [65].

In the scalable method proposed in [109], the approximation is controlled by two integer parameters: h, which removes the h least-significant rows, and v, which truncates the v partial product columns. Different (h, v) pairs define BAMs with different error-cost trade-offs.

In this thesis, we used the approximate multiplier generator provided by the authors of [109] to instantiate BAMs with various (h, v) settings. The goal was to evaluate whether these approximate multipliers could replace the exact ones in the Adaptive Quality Feature Extractor (AQFE) datapath in Figure 4.3, more precisely, in the Sparse Shift Multiplication Unit (SpSMU) and Sparse Reduction Unit (SpRU). The following subsections detail the algorithmic evaluation using the behavioral models and the hardware exploration with synthesis and power analysis.

D.2 Algorithmic Evaluation with Behavioral Models

We evaluated the impact of BAMs at the algorithmic level for each pair (h, v). We instantiated the approximate multipliers and used them as a drop-in replacements for the exact multipliers in a C model of the AQFE datapath. Features were recomputed on real Brain-Computer Interface (BCI) data using the same high-quality **(HQ)** configuration as in Chapter 3, and we measured the average Correlation Coefficient (CC) between features obtained with approximate and exact multipliers.

We first swept both parameters on a small grid to assess the relative effect of horizontal and vertical breaks. Even with no vertical truncation ($v = 0$), the CC degraded quickly as soon as h was increased. For $h = 1$ the average CC dropped to about 0.80, and for $h = 2$ it dropped further to 0.65. Given this rapid loss of correlation, we limited the exploration to $h = 0$ and only varied the vertical parameter v.

Figure D.2 shows the average CC as a function of v for $h = 0$. For $v \leq 13$, the correlation remains stable, above 0.99, and then slightly decreases, reaching around 0.97 at $v = 13$, and 0.92 at $v = 14$. Larger truncations yield faster degradation, with the CC reaching 0.42 at $v = 18$. These results indicate that moderate vertical truncation preserves a high correlation with the exact features, while aggressive truncation leads to distortions that are unlikely to be acceptable for the targeted decoding performance.

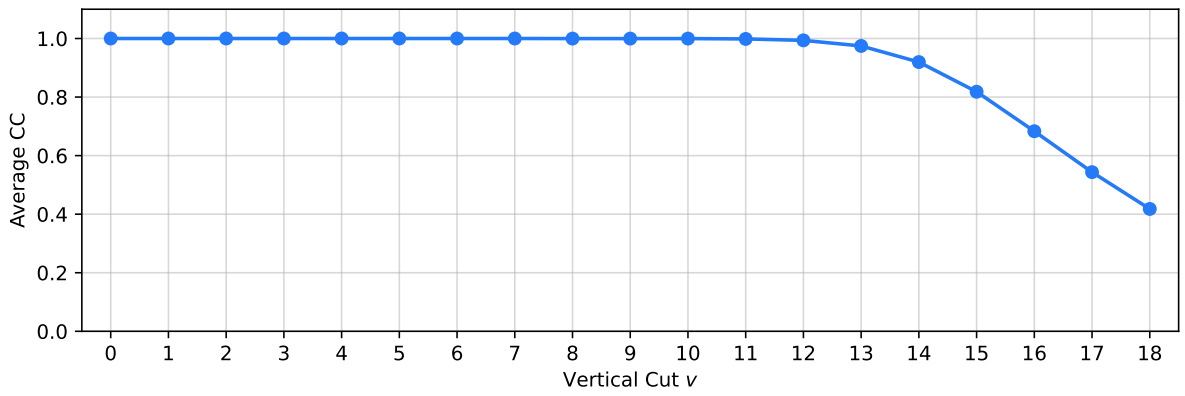


Figure D.2: Average CC vs. Vertical Cut Parameter v for BAM Multipliers with $h = 0$.

D.3 Hardware Exploration and Power Analysis

To complement the algorithmic tests, we explored the hardware cost of BAM using the same implementation flow as for the other blocks in this thesis. We select $h = 0$ and $v = 13$ as the horizontal and vertical cut parameters, respectively. Power estimates were then obtained using a PrimePower™ post-layout analysis with switching activity derived from real BCI data.

In principle, approximate multipliers should reduce logic complexity and switching activity, and thus dynamic power. However, in our experiments, the synthesized BAM implementations did not show a clear advantage over the exact multipliers available in the DesignWare™ library for the considered word lengths.

For the conservative configurations that preserved a high CC, the area and power of the approximate designs were worse than the highly optimized exact modules provided by Synopsys™. For more aggressive cuts, the implementation gains were not quantified as the CC drop did not justify their use.

Overall, in the context of this thesis, approximate BAMs did not provide a favorable trade-off between error and energy, especially when combined with the algorithmic-level approximations of the Piecewise Linear Continuous Wavelet Transform (PLCWT) and the custom bitwidth truncation. For this reason, they were not integrated into the final hardware design. This appendix therefore documents an exploratory path that was investigated but did not yield additional benefits for the targeted low-power BCI application.