Hardware/Software Co-Design and Reliability Analysis of Ultra-Low Power Biomedical Devices

Thèse N° 9231

Présentée le 12 avril 2019

à la Faculté des sciences et techniques de l'ingénieur
Laboratoire des systèmes embarqués
Programme doctoral en génie électrique

pour l'obtention du grade de Docteur ès Sciences

par

Soumya Subhra BASU

Acceptée sur proposition du jury

Prof. C. Guiducci, présidente du jury
Prof. D. Atienza Alonso, directeur de thèse
Prof. S. Mitra, rapporteur
Prof. L. Lavagno, rapporteur
Prof. C. Enz, rapporteur

2019

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Three things cannot be long hidden:
the Sun, the Moon and the truth.
— The Buddha

To my late maternal grandmother.
To my grandparents.
To my parents.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my PhD advisor, *Prof. David Atienza*, for his continuous support to my research over these past years, for his patience, motivation and immense knowledge. From him, I learned how to approach critical research problems, how to collaborate in a team, and how hard but efficient work can result in good research output. His guidance helped me both in my PhD research and in the writing of this thesis. I could not have imagined having a better advisor and mentor for my PhD.

I would like to take this opportunity to also thank my thesis jury members, *Prof. Carlotta Guiducci, Prof. Subhasish Mitra, Prof. Luciano Lavagno* and *Prof. Christian Enz*, for taking precious time out of their busy schedule to review this manuscript and provide insightful comments, which have encouraged me to widen my research from various perspectives. Additionally, my sincere thanks also goes to *Prof. Mitra*, who provided me with an opportunity to join the *Robust Systems Group* at *Stanford University* as an intern, giving me access to the laboratory and research facilities. Without his precious support, it would not be possible to complete this work. Furthermore, I would like to express my heartfelt thanks to *Prof. Laura Pozzi* for her boundless support and guidance throughout our fruitful collaboration with the Faculty of Informatics, *Università della Svizzera italiana (USI)*. There are, nevertheless, numerous other people who have helped me to produce this thesis, and in the rest of this section I would like to express my sincerest gratitude to them.

First of all, I would like to thank the postdoctoral researchers: *Giovanni, Pablo, Rubén, Miguel* and *Alexandre*, who have supervised my research work and given me valuable advice and inputs. I am also grateful for their infinite support during phases when I was stuck, which has helped me overcome various challenging situations during my PhD.

I cannot go on without acknowledging all the support of my colleagues at the *Embedded Systems Laboratory (ESL)* at EPFL, the people with whom I interacted on a daily basis. With them I have had significant sharing of knowledge, experience, and numerous chats over afternoon coffees, all of which have contributed directly or indirectly to the completion of this work. First of all, I would like to thank our secretaries *Homeira* and *Francine* for their precious support in organizing lab events and my trips to attend conferences in South Korea and many places in Europe, as well as my internship in California. I would like to seize this opportunity to thank our IT technician *Rodolphe* for all his behind-the-scenes support, and fast and efficient debugging of numerous IT-related issues that I have faced.

It was my honor to work (both directly and indirectly) with scientists and post-docs (past and

present) at EPFL and USI : *Hossein, Ivan, Francisco, Srinivasan, Shivani, Karim, Marina, Leila, Adriana, Shrikanth, Mohamed, Tomas* and *Jelena*. A special thanks goes to *Jelena* for showing me around USI and Lugano during my visits to the institute.

I would also like to specially acknowledge my colleague and friend, *Loris*, for our fruitful collaboration in the E4Bio project during our PhD periods at ESL, and with whom I have also done many activities outside of work.

In addition, I would also like to thank my past and present officemates: *Ivan, Arman, Marina* and *Amir* for creating such a nice working atmosphere at ELG 135. A special thanks to *Arman* for being a very nice travel companion during our trip to Seoul for attending ESWEEK 2017.

I'd also like to thank my fellow PhD students at ESL: *Ali, Grégoire, Artem, Fabio, Dionisije, Elisabetta, Yasir, Alessandro* and *Luis* for all the sharing of ideas and many interesting discussions. Finally, I'd like to wish the best to the new generation of PhD students and members at ESL, who I am sure will continue the legacy and reputation of our laboratory: *Farnaz, Halima, Benoit, Andrew, Kawsar, Lara, Szabolcs, Wellington, Damián, José, Renato, Fabio* and *Victoriano*.

Furthermore, I'd like to thank my colleagues and friends from other labs at EPFL/Switzerland/elsewhere: *Francesca S. & Enrico, Francesca C., Irene, Ioulia, Ivan, Giovanni, Tugba, Sofia, Nadia, Eleonora, Francesco, Maneesha, Emanuele, Nicolo, Erick, Wolfgang, Mariazel, Gianrocco, Evgeniy, Lana, Wiola, Harshal, Manuel, Matteo, Joanna, Vitalijs* and *Merve* for all the good times we have shared together.

In parallel to my work, I have had the privilege to meet and be friends with a number of people, both at EPFL and outside, who have shaped my life in the past few years. In particular, I'd like to thank my close Bengali friends: *Arnab, Shankha* and *Abhirup* for all the trips, dinners, chats and what not. A very special thanks goes to my Italian "famiglia": *Ylenia, Antonella* and *Federico*, and to my Swiss best friends: *Christelle* and *Joel* for making me feel at home whenever I visited them, as well as for their endless encouragement. Finally, my heartfelt thanks goes to *Agata*, for all her support and for listening patiently to all the stories, experiences and troubles that I've shared with her.

I am also grateful to the Indian students' association of Lausanne, *YUVA*, for which I have had the privilege to serve as the treasurer in 2014-2015, and as the president in 2015-2016 academic years. I would especially like to thank *Abhishek, Raja, Teju, Preeti* and *Elahi* for imparting their experience and for being members of a highly efficient core committee. I wish the association a long life and all the best for the new members running it.

Last but not the least, there are no words to describe my gratitude towards all the members of my family for their unconditional love, inspiration and encouragement. I'd particularly like to thank my father, my mother, my maternal grandfather and my (late) maternal grandmother for being such strong pillars of support since my early childhood till now, guiding me through thick and thin.

Lausanne, 20th December 2018 Soumya Subhra Basu

Abstract

Smart health monitoring devices, known as **Wireless Body Sensor Nodes (WBSN)**, are transforming today's healthcare landscape, shifting it from traditional hospital-based methods toward more personalized approaches. Their demand is ever-increasing in the modern world, where an overwhelming majority of deaths are caused due to chronic disorders, especially by cardiovascular diseases. WBSNs provide low-cost, unobtrusive and wearable solutions for continuous health-monitoring. In addition to reducing healthcare costs drastically, they improve the quality of life of monitored patients. These small autonomous devices are capable of acquiring, processing and wirelessly transmitting biological signals to medical personnel, typically relying on a limited on-board power source. Since WBSNs are desired to operate over prolonged periods, producing potentially critical data, their reliability and energy efficiency are of paramount importance. These characteristics call for optimizations in the software running on the WBSNs as well as in the underlying hardware.

State-of-the-Art (SoA) WBSNs are able to process on-board the sensed bio-signals, like Electrocardiograms (ECG), transmitting only compact sets of clinically-relevant features instead of the entire acquired signal. This scenario moves the energy bottleneck of such devices from the transmission link to the computation-intensive Bio-Signal Processing (BSP) segment. To improve their energy efficiency, in this thesis, I propose strategies ranging from architectural to technology levels.

First, I explore the applicability of a **relaxed-reliability** computation paradigm, resulting from ultra-low voltage operations, in the BSP domain. I begin by modeling memory failures ensuing from voltage over-scaling and fabrication variability-related issues, studying their effects at the application level. Subsequently, I analyze reliability-aware methods for WBSNs, both at the memory and system levels, that drastically decrease their energy consumption while safeguarding proper system operation. To this end, I introduce novel **data significance-based** protection approaches for countering failures in embedded memories, showcasing up to 20 % reduction in the energy budget compared to existing solutions. In addition, I devise system-wide **lightweight hardware- and software-level error monitoring solutions** that cope with failures in the entire WBSN memory subsystem, waiving their protection requirement. Experimental results demonstrate that these strategies ensure minimal performance degradation, while at the same time avoiding system crashes, subsequently achieving up to 24 % energy savings with respect to the SoA.

In the second part of the thesis, I employ novel energy-efficient approaches featuring **heteroge-**

neous platforms with a reconfigurable accelerator and **emerging memory technologies**, to improve the efficiency of traditional WBSN architectures. I commence by performing a study to identify the computationally intensive segments of BSP algorithms that can potentially be accelerated on a reconfigurable fabric. I adopt a top-down approach to do so, extracting kernels from the high-level application code, eventually optimizing and mapping them for execution on a time- and resource-shared Coarse Grain Reconfigurable Array (CGRA). I prove the effectiveness of the proposed platform by exhibiting up to 11x performance improvement and 37 % energy savings over the SoA. Then, I **combine relaxed-reliability computation and CGRA-based heterogeneous WBSNs**, exploiting the attainable benefits when both strategies are jointly leveraged. My experiments exhibit as much as 70 % reduction in energy consumption compared to the current SoA, while assuring safe system operation. I conclude this work by exploring the integration of emerging **Resistive Random Access Memory (ReRAM)** technologies to traditional WBSN platforms, forming heterogeneous hierarchies aimed at reducing their ever-increasing leakage energy consumption. In this regard, I report the memory-level endurance challenges faced due to aging, and propose lightweight solutions for prolonged device usage. Obtained results highlight up to 65 % increase in the energy efficiency with respect to the SoA, including the overheads required for extended device functionality.
My work shows that the high-level characteristics of bio-signal applications (in particular, the tolerance toward a small degree of quality degradation and the prominence of computational hot-spots at runtime), and emerging memory systems can be effectively exploited to significantly increase the energy efficiency of WBSN devices, paving the way for deeply embedded and smart health monitors.

Keywords: Bio-signal processing, Ultra-low power, Relaxed-reliability computation, Voltage over-scaling, Inexact computation, Error monitoring, Data significance, Memory protection, Supply voltage droops, Heterogeneous architecture, Coarse grain reconfigurable array, Computational kernel, ReRAM endurance.

Résumé

Les dispositifs intelligents de surveillance de la santé, appelés **Wireless Body Sensor Nodes (WBSN)**, transforment le paysage actuel des soins de santé, passant des soins hospitaliers traditionnels à des approches plus personnalisées. Leur demande ne cesse d'augmenter dans le monde moderne, où une majorité écrasante des décès sont dus à des troubles chroniques, notamment des maladies cardiovasculaires. Les WBSNs fournissent des solutions portables, discrètes et peu coûteuses pour une surveillance continue de l'état de santé. En plus de réduire considérablement les coûts des soins de santé, ils améliorent la qualité de vie des patients suivis. Ces appareils autonomes et à faible encombrement sont capables d'acquérir, de traiter et de transmettre sans fil des signaux biologiques au personnel médical, en utilisant généralement une source d'énergie embarquée limitée. Étant donné que les WBSNs sont appelés à fonctionner sur des périodes prolongées, produisant des données potentiellement critiques, leur fiabilité et leur efficacité énergétique sont d'une importance primordiale. Ces caractéristiques nécessitent à la fois des optimisations dans les parties logicielles et matérielles des WBSNs.

Les WBSNs à la pointe de la technologie sont capables de traiter les signaux biologiques acquis, tels que les électrocardiogrammes (ECG), en ne transmettant que des ensembles compacts de caractéristiques pertinentes sur le plan clinique au lieu de l'intégralité du signal brut. Ce scénario déplace le goulot d'étranglement énergétique de ces périphériques de la transmission vers le traitement du signal biologique (Bio-Signal Processing, BSP), qui nécessite beaucoup de calculs. Pour améliorer leur efficacité énergétique, dans cette thèse, je propose des stratégies allant de l'architecture aux niveaux technologiques.

Premièrement, j'explore l'applicabilité d'un paradigme de calcul de **fiabilité relâchée** résultant d'opérations à très basse tension dans le domaine du BSP. Je commence par modéliser les défaillances de mémoire résultant de problèmes de surdimensionnement de la tension et de la variabilité de la fabrication, en étudiant leurs effets au niveau de l'application. Par la suite, je conçois des méthodes qui tiennent compte de la fiabilité pour les WBSNs, à la fois au niveau de la mémoire et du système, et qui réduisent considérablement leur consommation énergétique tout en préservant le bon fonctionnement du système. À cette fin, j'introduis de nouvelles techniques de protection des mémoires **basées sur l'importance des données** pour contrer les défaillances dans les mémoires intégrées, en présentant une réduction jusqu'à 20 % du budget énergétique par rapport aux solutions existantes. En outre, je conçois au niveau du système des **solutions légères de surveillance des erreurs matérielles et logicielles**, qui

permettent de faire face aux erreurs dans les mémoires et renoncent à l'exigence de protection. Les résultats expérimentaux montrent que ces stratégies garantissent une dégradation minimale des performances tout en évitant les défaillances du système, réalisant jusqu'à 24 % d'économies d'énergie par rapport à l'état de l'art.
Dans la seconde partie de la thèse, j'utilise de nouvelles approches d'économie d'énergie utilisant des plateformes **hétérogènes avec un accélérateur reconfigurable** et **des technologies de mémoire émergentes**, afin d'améliorer l'efficacité des architectures WBSNs traditionnelles. Je commence par effectuer une étude pour identifier les segments d'algorithmes BSP à calcul intensif qui peuvent potentiellement être accélérés sur une structure reconfigurable. J'adopte une approche descendante pour ce faire, en identifiant les noyaux du code d'application de haut niveau, pour finalement les optimiser et les assigner pour qu'ils soient exécutés sur un accélérateur reconfigurable à gros grains (Coarse Grain Reconfigurable Array, CGRA) à temps et ressources partagées. Je démontre l'efficacité de la plateforme proposée, mettant en avant des performances de calculs jusqu'à 11 fois supérieurs, tout en offrant 37 % d'économie d'énergie par rapport aux meilleures solutions existantes. Enfin, je **combine le calcul de fiabilité relaxée et les WBSN hétérogènes basés sur CGRA**, en exploitant les avantages réalisables lorsque les deux stratégies sont mises en commun. Mes expériences démontrent une réduction de 70 % du budget énergétique par rapport à l'état de l'art actuel, tout en assurant un fonctionnement fiable du système. Je conclus cette étude en explorant l'intégration des technologies émergentes de **Resistive Random Access Memory (ReRAM)** aux plateformes WBSN traditionnelles, formant des hiérarchies hétérogènes visant à réduire leurs fuites énergétiques toujours croissantes. Pour cela, je mets en évidence les problèmes d'endurance liés au vieillissement des mémoires, et je propose des solutions légères pour une utilisation prolongée de l'appareil. Les résultats obtenus mettent en évidence une augmentation de l'efficacité énergétique de l'ordre de 65 % par rapport à l'état de l'art, y compris les frais requis pour la fonctionnalité étendue de l'appareil.
Mon travail montre que les caractéristiques de haut niveau des applications biomédicales (en particulier la tolérance à un faible degré de dégradation de la qualité et l'importance des noyaux de calculs intensifs au moment de l'exécution) et les systèmes de mémoire émergents peuvent être efficacement utilisés pour augmenter significativement l'efficacité énergétique des WBSNs, ouvrant la voie à des dispositifs de suivi médical intégrés et intelligents.

Mots-clés : Traitement des signaux biologique, Très faible consommation, Calcul de fiabilité relâchée, Dépassement d'échelle de tension, Calculs inexacts, Surveillance des erreurs, Signification des données, Protection de la mémoire, Chute de tension d'alimentation, Système hétérogène, Accélérateur reconfigurable à gros grains, Endurance de ReRAM.

Contents

Acknowledgements v

Abstract (English/Français) vii

List of Figures xiii

List of Tables xviii

List of Acronyms xx

1 Introduction 1
- 1.1 Wireless body sensor nodes . . . 2
  - 1.1.1 WBSN functionalities . . . 3
  - 1.1.2 Smart WBSNs . . . 4
- 1.2 Energy bottlenecks and optimizations in WBSNs . . . 8
  - 1.2.1 Energy bottlenecks . . . 8
  - 1.2.2 Optimization: voltage supply scaling and management . . . 9
  - 1.2.3 Optimization: domain-specific architectures . . . 12
- 1.3 Contributions of this thesis . . . 13
  - 1.3.1 Relaxed-reliability computation for ultra-low power bio-signal processing 14
  - 1.3.2 Heterogeneous architectures for ultra-low power WBSN platforms . . . 15
- 1.4 Thesis Outline . . . 17

2 Relaxed-Reliability Computation for Ultra-Low Power Bio-Signal Processing 19
- 2.1 Introduction . . . 19
  - 2.1.1 Energy concerns in WBSNs . . . 20
  - 2.1.2 State-of-the-art . . . 22
    - 2.1.2.1 Hardware- and technology-based memory protection . . . 24
    - 2.1.2.2 Hardware/software-based error mitigation . . . 25
  - 2.1.3 Contributions and outline of the chapter . . . 27
- 2.2 Relaxed-reliability computation with significance-based memory protection . 28
  - 2.2.1 Introduction . . . 28
  - 2.2.2 Ultra-low voltage scaling in the data memory . . . 29
    - 2.2.2.1 Data-bit significance-based protection for non-sparse buffers . 29

2.2.2.2 Word significance-based protection for sparse buffers 30
2.2.2.3 Experimental setup and results 31
2.2.3 Extension of ultra-low voltage operation to the entire WBSN system . . . 39
2.2.3.1 Experimental setup and results 40
2.3 Relaxed-reliability computation with error management 46
2.3.1 Introduction 46
2.3.2 Effects of voltage scaling on SRAM memories in presence of supply fluctuations 47
2.3.2.1 Memory failure modeling 49
2.3.2.2 Application-level error manifestation 50
2.3.3 Experimental setup and results 52
2.3.3.1 Target evaluation bio-signal processing platform 52
2.3.3.2 Error monitoring and recovery strategies 55
2.3.3.3 Experimental results 59
2.4 Summary of contributions 71

3 Heterogeneous Architectures for Ultra-Low Power WBSN Platforms 73
3.1 Introduction 73
3.1.1 State-of-the-art 74
3.1.1.1 Domain-specific optimizations 74
3.1.1.2 Reconfigurable architectures 77
3.1.1.3 WBSN design using non-volatile memories 78
3.1.2 Contributions and outline of the chapter 79
3.2 Reconfigurable bio-signal processing architectures 81
3.2.1 Introduction 81
3.2.2 Single Instruction Multiple Data (SIMD) CGRA-based bio-signal processing 82
3.2.2.1 Basic single-datapath CGRA structure 82
3.2.2.2 CGRA with SIMD processing 86
3.2.2.3 Reconfigurable WBSN with SIMD CGRA 88
3.2.2.4 Kernel mapping on the SIMD CGRA 90
3.2.2.5 Experimental setup and results 96
3.2.3 Interleaved Datapaths (i-DPs) CGRA-based bio-signal processing 107
3.2.3.1 SIMD CGRA limitations 107
3.2.3.2 i-DPs CGRA structure 108
3.2.3.3 Kernel mapping on the i-DPs CGRA 109
3.2.3.4 Experimental setup and results 110
3.2.4 Reconfigurable WBSNs with relaxed-reliability computation 115
3.2.4.1 Heterogeneous and relaxed-reliability computation 115
3.2.4.2 Experimental setup and results 116
3.3 Resistive RAM-based heterogeneous WBSN architectures 123
3.3.1 Introduction 123
3.3.2 Emerging ReRAM memories 123

3.3.3 Experimental setup . . . 125
3.3.4 Experimental results . . . 129
3.3.4.1 Framework . . . 129
3.3.4.2 Obtained results . . . 131
3.4 Summary of contributions . . . 135

4 Conclusions and Future Work **137**
4.1 Summary and contributions . . . 137
4.2 Future Work . . . 139

A Reliability analysis of ReRAM-based heterogeneous WBSN memory featuring SRAM page buffers **143**

Bibliography **160**

Curriculum Vitae **161**

List of Figures

1.1 Proportion of causes of deaths worldwide under the age of 70. 1
1.2 Healthcare costs as a percentage of the GDP. 2
1.3 A wireless body sensor network. A multi-lead ECG processing WBSN is shown in the bottom-left corner (highlighted in red). 3
1.4 Schematic diagram of a smart WBSN system. 4
1.5 Energy breakdown of traditional WBSNs (left) and smart WBSNs (right). 5
1.6 Schematic diagram of a human ECG representing a single heartbeat. 6
1.7 Example of a block scheme of a smart WBSN application. In addition to signal acquisition and transmission, these devices perform BSP on-board to extract the important features. 7

2.1 A block scheme of a typical WBSN, highlighting the BSP segment where relaxed-reliability computation paradigm can be applied. 23
2.2 ECC-based protection for non-sparse buffers. 29
2.3 ECC- and parity-based protection for sparse buffers. 30
2.4 Block-scheme of the PSA system. 33
2.5 Statistical distribution of the data in the different buffers: a) *Extr_buffer* and b) *DWT_buffer*. 34
2.6 Percentage of error in the calculation of the LFHF ratio under the different protection schemes: a) at 650 mV and b) at 600 mV. 35
2.7 Total energy consumption at 650 mV supply under the different memory protection schemes for an execution time ≈ 2.67s. 37
2.8 Percentage error in the LFHF ratio and the corresponding energy consumption under the different studied memory protection schemes. Numbers beside the points represent the percentage of *DWT_buffer* protected. 38
2.9 Block scheme of a typical WBSN's processing unit, representing the components active during bio-signal processing. 40
2.10 The proposed heterogeneous protection scheme applies varying exactness guarantees depending on the data criticality. 41
2.11 Percentage error in the LFHF ratio under the different memory protection schemes: a) at 650 mV and b) at 600 mV. 42
2.12 Total system energy consumption at baseline 1 (high V_{dd}). 44

2.13 Total system energy consumption at baseline 2 (low V_{dd} with complete ECC protection). . . . 44
2.14 Total energy consumption by the targeted memory buffers at 650 mV supply under different memory protection schemes for an execution time of ≈ 2.23 s. . 45
2.15 Percentage error in LFHF ratio and the corresponding energy consumption under different memory protection schemes at 650 mV. Numbers beside the points represent the percentage of *DWT_buffer* protected. . . . 46
2.16 The developed framework analyzes the impact of hardware failures (top) at the application level, when executing on the proposed error-resilient platform (bottom). . . . 48
2.17 Error rates at nominal voltage (right), dependent on static variability (left, gray area) and voltage noise profile (left, blue line). . . . 49
2.18 A flowchart showing the application level error classification strategy. . . . 51
2.19 Block diagram of the TamaRISC architecture. . . . 53
2.20 The target bio-signal processing platform. . . . 54
2.21 The proposed error classification strategy along with the corresponding action to be taken in each case. . . . 55
2.22 The target bio-signal processing platform with the proposed error-monitoring strategies. . . . 57
2.23 Baseline wander in ECG. In blue: acquired raw ECG with baseline wander, in orange: ECG with baseline wander removed. . . . 61
2.24 Block diagram showing the mapping of highly parallel CS workloads on multiple processors. . . . 62
2.25 Block diagram showing the mapping of MF-CS workloads on the multiple processors, exhibiting a producer-consumer relationship. . . . 62
2.26 Failure probabilities per bit in the instruction and data memory banks for 100 variation maps considering a fixed supply voltage value (a, c), and in the presence of supply voltage droops (b, d). . . . 63
2.27 Classification of the resulting application-level errors. . . . 65
2.28 Runtime error profile, considering the access frequency of memory words. . . . 66
2.29 Example of signals impacted by SDC errors relative to an error-free ECG: a) the expected ECG signal, b) example of corrupted ECG accepted at the receiver side, c) example of a rejected ECG window. . . . 67
2.30 Evolution of the Quality-of-Service (QoS) for different voltage supply values and droop variations. . . . 69
2.31 Evolution of the energy consumption per correct output (compressed sample) produced, for different voltage supply values and droop variations. . . . 69
2.32 Power consumption comparison of the error-free multi-core system (operating at 1.27 V) with respect to the proposed inexact system (operating at 1.13 V), which guarantees a QoS of over 99 % for both applications. . . . 70

3.1 The proposed heterogeneous platform featuring a CGRA accelerator shared by the multi-processors. . . . 81
3.2 Block scheme of the shared CGRA featuring a single datapath per cell, with one row of the RCs implementing square root calculator (denoted in red). The configuration RAM (also part of the CGRA) is not shown in this figure. . . . 82
3.3 RC architecture of the single-DP CGRA, highlighting the details of the datapath. The CGRA configuration RAM is not shown for clarity. . . . 84
3.4 Block scheme of the SIMD CGRA, with one row of the PEs implementing square root calculator (denoted in red). Outputs and flags from each DP are routed to the corresponding DPs in the neighboring RCs. . . . 87
3.5 RC architecture of the SIMD CGRA, highlighting a) the datapath structure, and b) the format of the configuration words. All DPs in the RC receive the same configuration word at each clock cycle. . . . 88
3.6 High-level view of the proposed bio-signal processing platform. . . . 89
3.7 Detailed example of mapping of the *CS* kernel on the CGRA. . . . 94
3.8 Block scheme of the experimental framework, comprising RTL and cycle-accurate system views. . . . 97
3.9 Acquired ECG signal (top) and its MMD-delineated version (bottom). . . . 99
3.10 Average speed-up of kernels running on the multi-core + CGRA platform, compared to execution only on the multi-core platform. . . . 102
3.11 Energy consumption of the different kernels (for the employing application). For each kernel, the bars are normalized to the SW-only consumption, which is indicated on the right side of the graph. . . . 104
3.12 System energy consumption for processing the different benchmark applications. 105
3.13 Area breakdown of proposed platform, embedding a CGRA with different SIMD widths. . . . 106
3.14 The i-DPs CGRA block scheme, interfaced with a multiprocessor system. . . . 108
3.15 Example of a simple kernel mapped in a single-DP and an i-DPs CGRA. In the latter case, the kernel is split into two slices and mapped on the interleaved DPs of the RCs. . . . 109
3.16 Runtime of *Dbl Min Srch* kernel (first and second minimum element of an array), on a single-DP CGRA and on i-DPs with different widths, varying the input data size. . . . 111
3.17 Average kernels runtime (in clock cycles) executing on CGRAs with 1 DP and 2 i-DPs. . . . 112
3.18 Energy consumption of the different kernels. For each kernel, the bars are normalized to the SW-only energy. . . . 113
3.19 System energy consumption of the different benchmark applications, normalized to the SW-only consumption for the part of the application transfered to the CGRA. . . . 113
3.20 Area breakdown of the i-DPs CGRA compared to a traditional 1 DP CGRA. . . . 114

3.21 Previous works have explored, in isolation, multi-core processing, CGRA acceleration and inexact computing in the field of ultra-low power bio-signal processing. In this section, I explore their ensuing synergies. 115
3.22 High-level block scheme of the proposed heterogeneous architecture with relaxed-reliability requirements. 117
3.23 Quality-of-Service (QoS) for different voltage supply values and droop variations for the proposed architecture. 121
3.24 Evolution of the energy consumption per compressed sample produced, for different voltage supply values and droop variations. 121
3.25 Detailed energy consumption breakdown for the studied system. 122
3.26 Reliability of a commercial 64 KiB ReRAM module. After $\approx 1.8 \times 10^6$ writes per word, all the words in the chip are damaged. A word is considered damaged when at least one bit in it is flipped and stuck at the changed value. 125
3.27 Programming operations in ReRAM technologies (*set* in red and *reset* in gray) with detailed voltage current and energy considerations. ReRAM corners (fast, slow) are also defined. 126
3.28 Schematic representation of the heterogeneous architecture. 127
3.29 Write accesses distribution over data words for CS. Vertical bars show the grouping of words into 64-word blocks, with the maximum number of writes to any word in each block. Also shown (orange) is the distribution of accesses if the application variables can be reordered. 129
3.30 The proposed ReRAM block relocation mechanism (maximum 256 MB addressable) embedded in the MMU. 130
3.31 Average power of the system components at 1 V supply. 132
3.32 WBSN energy consumption embedding SRAM (blue) or ReRAM (red) against ReRAM programming energy. Energy ratio between SRAM and ReRAM based WBSNs in gray. 133
3.33 ReRAM over-provisioning requirements to achieve up to one year of system lifetime, depending on the number of writes to a word before it fails permanently. 134

A.1 Run-time error profile comparison of the studied architecture against an SRAM-only one, considering the access frequency of memory words. 143
A.2 Comparison of the Quality-of-Service (QoS) evolution for the two architectures for different voltage supply values and droop variations. The dotted line represents the condition with SRAM page buffers. 144

List of Tables

2.1 Number of SECDED ECC bits required to protect data bits. 30
2.2 Area of the multi-core platform and of its architectural components. 68

3.1 Kernels utilization per application. Each cell indicates the number of processing cores requesting each acceleration. 100
3.2 CGRA area exploration (in 65 nm UMC technology library). 106
3.3 Comparison of various memory parameters for SRAM and ReRAM. 124
3.4 Energy consumption per access and leakage power for each type of memory used in the different platform configurations. 128

List of Acronyms

1T1R 1-Transistor 1-Resistor

ADC Analog-to-Digital Converter

AI Artificial Intelligence

ALU Arithmetic Logic Unit

ASIC Application-Specific Integrated Circuit

BRT Block Relocation Table

BSP Bio-signal Processing

CGRA Coarse Grain Reconfigurable Array

CMOS Complementary Metal-Oxide-Semiconductor

CR Configuration Register

CS Compressed Sensing

DM Data Memory

DMA Direct Memory Access

DP Datapath

D-PB Data Page Buffer

DSIP Domain Specific Instruction-set Processor

DSP Digital Signal Processing

DWT Discrete Wavelet Transform

ECC Error Correction Code

ECG Electrocardiogram

EEG Electroencephalography

eFlash Embedded Flash Memory

EM Execution Monitor

EMG Electromyogram

FFT Fast Fourier Transform

FIFO First-In-First-Out

FPGA Field Programmable Gate Array

HRS High Resistance State

IC Integrated Circuit

II Initiation Interval

IM Instruction Memory

IoT Internet of Things

ITRS International Technology Roadmap for Semiconductors

LFSR Linear Feedback Shift Register

LIFO Last-In First-Out

LRS Low Resistance State

LSB Least Significant Bit(s)

MF Morphological Filtering

MIMD Multiple Instruction Multiple Data

MMD Morphological Derivative-based Delineation

MMU Memory Management Unit

MSB Most Significant Bit(s)

NCD Non-Communicable Disease

NOP No Operation

NTV Near-Threshold Voltage

NVM Non-Volatile Memory

OxRAM Oxide-based Random Access Memory

PDA Personal Digital Assistant

PE Processing Element

PID Processor Identifier

P-PB Program Page Buffer

PRD Percentage Root-mean-square Difference

PSA Power Spectral Analysis

QoS Quality-of-Service

RC Reconfigurable Cell

ReRAM Resistive Random Access Memory

RF Register File

RISC Reduced Instruction Set Computer

RMS Root Mean Square

RP-CLASS Random Projection-based Classification

SCM Standard Cell-based Memories

SDC Silent Data Corruption

SECDED Single Error Correction and Double Error Detection

SIMD Single Instruction Multiple Data

SNM Static Noise Margin

SNR Signal-to-Noise Ratio

SoA State-of-the-Art

SoC System-on-Chip

SRAM Static Random Access Memory

SW Software

TSV Through-Silicon Via

VFS Voltage-Frequency Scaling

WBSN Wireless Body Sensor Node

WFFT Wavelet-based Fast Fourier Transform

WHO World Health Organization

1 Introduction

Studies by the World Health Organization (WHO) show that Non-Communicable Diseases (NCDs) are responsible for more than half of all deaths globally (40.5 million in 2016) [1]. NCD refers to a medical condition that is not caused by infectious agents, but rather by diseases which last for long periods of time and progress slowly. WHO's studies also show that every year 15 million people between the ages of 30 and 69 die from an NCD, with over 80 % of these deaths occurring in low- and middle-income countries [2]. NCDs typically include cardiovascular disorders, cancers and respiratory diseases, the risks of dying from which are increased by tobacco and alcohol use, lack of physical activities and unhealthy diets. Among them, Cardiovascular Diseases (CVD) account for more than a third of all NCD-related deaths (Figure 1.1) [1], thus making them a critical subject in medical research.

In today's world, busy and unhealthy lifestyles are becoming common, thereby resulting in an increase in the number of people with CVDs. Moreover, a significant part of the world population is aging: currently more than 962 million people are aged 60 or more, with this number expected to double over the next few decades [3]. Therefore, in the near future, a large portion of the population of the world will become susceptible to contracting cardiac diseases.

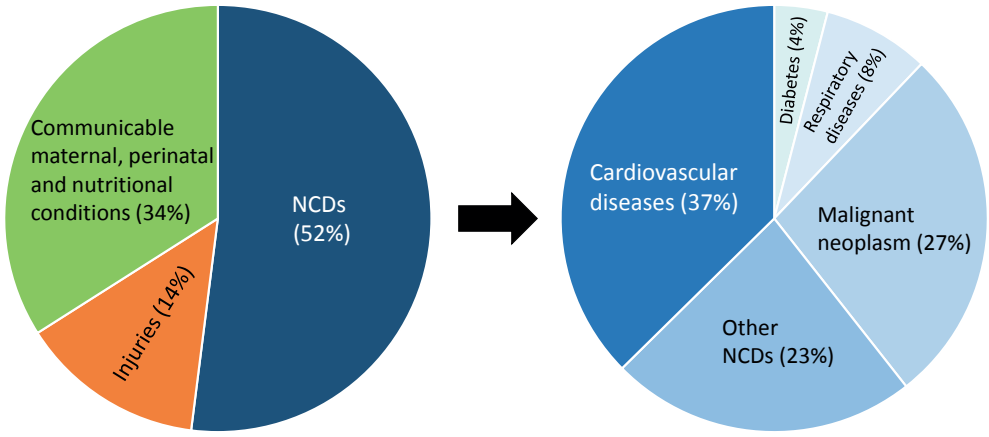


Figure 1.1 – Proportion of causes of deaths worldwide under the age of 70.

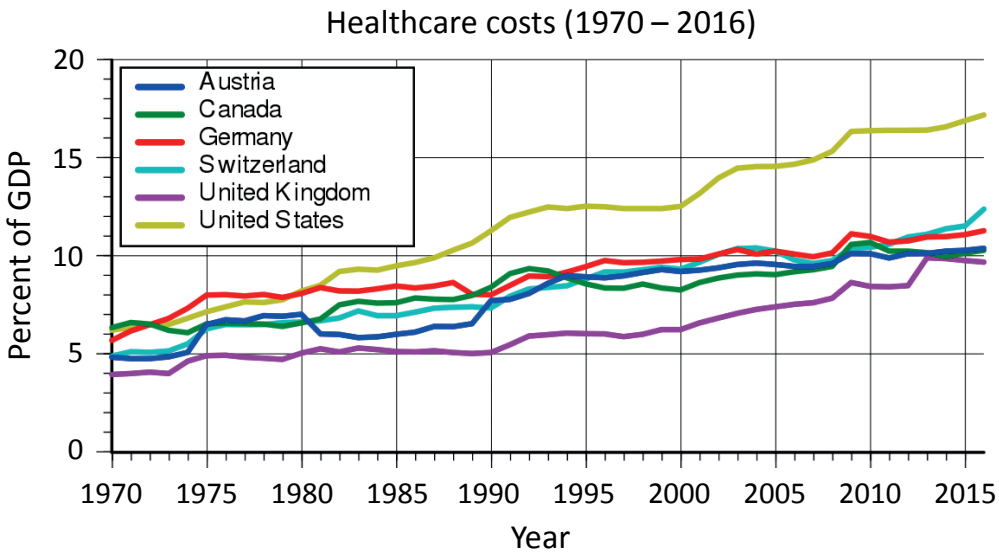


Figure 1.2 – Healthcare costs as a percentage of the GDP.

This serious scenario calls for increased levels of close and continuous medical supervision and management. Traditional healthcare for NCDs, and especially for cardiac diseases, is typically hospital-based. This form of healthcare involves regular and often long hospital visits by patients, where the number of staff and medical supervisors is limited. Furthermore, the costs incurred in hospital-based health monitoring are running in the order of hundreds of billion dollars and increasing (Figure 1.2) [4] [5]. Thus, there is an urgent need for a paradigm shift in health monitoring, moving toward a more personalized approach, where prevention and early diagnosis of diseases are emphasized in order to bring down expenses.

1.1 Wireless body sensor nodes

Wireless body sensor network technologies can offer large-scale, low-cost and unobtrusive solutions to perform continuous health-monitoring [6]. Wearable Internet of Things (IoT) devices, known as Wireless Body Sensor Nodes (WBSNs), are the building blocks of such a network (Figure 1.3) [7] [8]. These embedded-system devices offer long-term health supervision outside of a hospital environment, providing real-time and personalized monitoring. As a result, they are instrumental in minimizing the requirement of surveillance form medical staff. WBSNs are responsible for the cooperative monitoring of various health indicators, and are generally coordinated using a central node.

These miniaturized battery-powered devices provide wireless and wearable solutions that are generally ambulatory. Thus, WBSNs decrease patients' discomfort compared to traditional systems, where sensors are usually attached to their bodies and are connected with wires to bulky analyzing equipment. As a result, patients can be monitored while performing daily activities, and the acquired data can be analyzed remotely by medical personnel. This helps in avoiding (often) unnecessary hospital visits and the related costs. In case of an

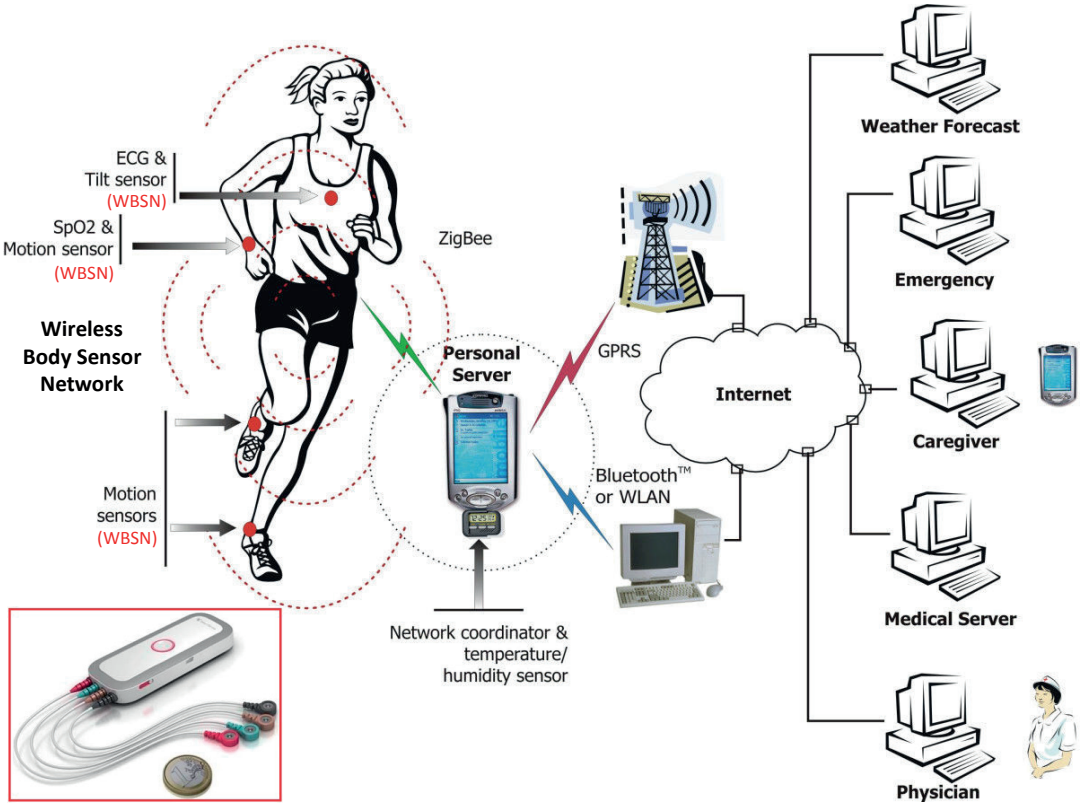


Figure 1.3 – A wireless body sensor network. A multi-lead ECG processing WBSN is shown in the bottom-left corner (highlighted in red).

anomaly detected by analyzing the monitored data, the medical personnel can be alerted for intervention.

1.1.1 WBSN functionalities

WBSNs are capable of sensing and acquiring biological signals like Electrocardiograms (ECG), peripheral oxygen saturation (SpO_2), Electromyograms (EMG), behavioral information (e.g., movement) and environmental data (e.g., temperature, humidity, light) [9] [10]. For increasing the accuracy of the acquired data, WBSNs generally need to employ multiple sensors to capture bio-signals from different parts of the body [11]. In addition, these devices (an example of which is shown in the bottom-left corner of Figure 1.3) are also able to transmit the acquired data over wireless links to a central data collector like a Personal Digital Assistant (PDA). PDAs, in turn, send the data (possibly over the Internet) to doctors or hospital staff, for further analysis.

Such data exchanges, however, involve power-hungry transmitters on WBSNs, and the analysis of the received sensed data generally consists of labor-intensive manual inspection or off-line execution on a server infrastructure. Also, when entire lengths of the acquired bio-signals

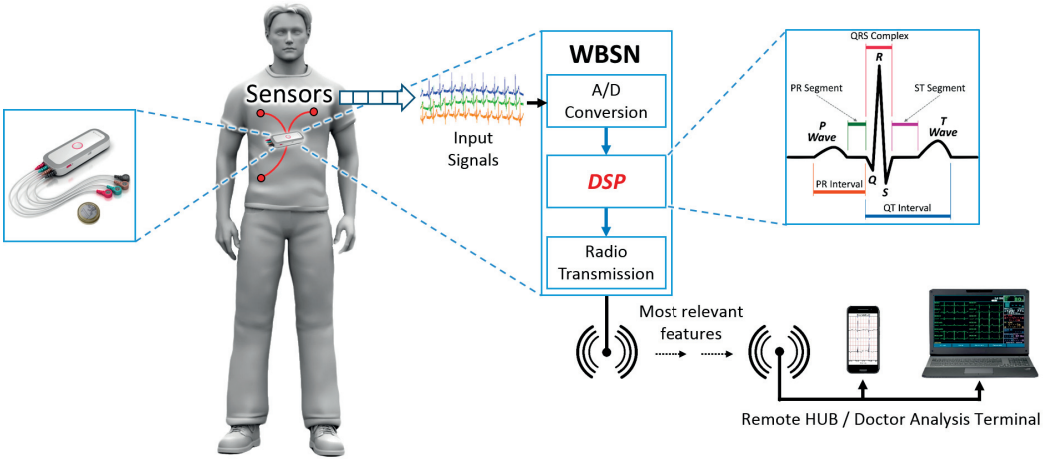


Figure 1.4 – Schematic diagram of a smart WBSN system.

are transmitted, WBSNs are required to manage and send a large volume of data. The high expense of data transmission stresses the limited on-board power source, thereby making energy a very critical resource for WBSNs. In addition, to minimize the discomfort of patients using them, the dimensions of WBSNs must be small, hence restricting the on-board battery size and capacity, therefore calling for more energy-efficient solutions.

1.1.2 Smart WBSNs

A popular approach to reduce energy consumption in WBSNs is to perform *feature*-based processing in order to limit the amount of data to be transmitted. To this end, a set of clinically important parameters (termed as *features*) are extracted from the acquired bio-signals, that are relevant for the target application. For example, WBSNs for ECG process the acquired signals to extract and transmit the intervals between heartbeats, instead of transmitting the entire ECG signal.

In this context, a new generation of energy-efficient *smart* WBSNs has emerged [12], which are capable of performing Digital Signal Processing (DSP), also called *Bio-signal Processing (BSP)*, of the acquired bio-signals on the device itself. By doing so, they extract the relevant signal features, before transmitting them wirelessly (Figure 1.4). These devices are generally composed of bio-signal sensors, embedded processors of limited computational capacity, on-board memory subsystems and wireless transmission systems.

BSP applications running on these smart WBSNs usually execute moderately complex algorithms that condition the acquired signals (e.g., noise cancellation, signal compression), and extract the features (such as heartbeat rate in case of ECG). The extracted features are then further processed to interpret them clinically and potentially perform an early analysis for disease detection. By doing the DSP on-board, smart WBSNs significantly increase the energy efficiency, and in turn the battery lifetime [13]. In fact, it has been shown that by performing

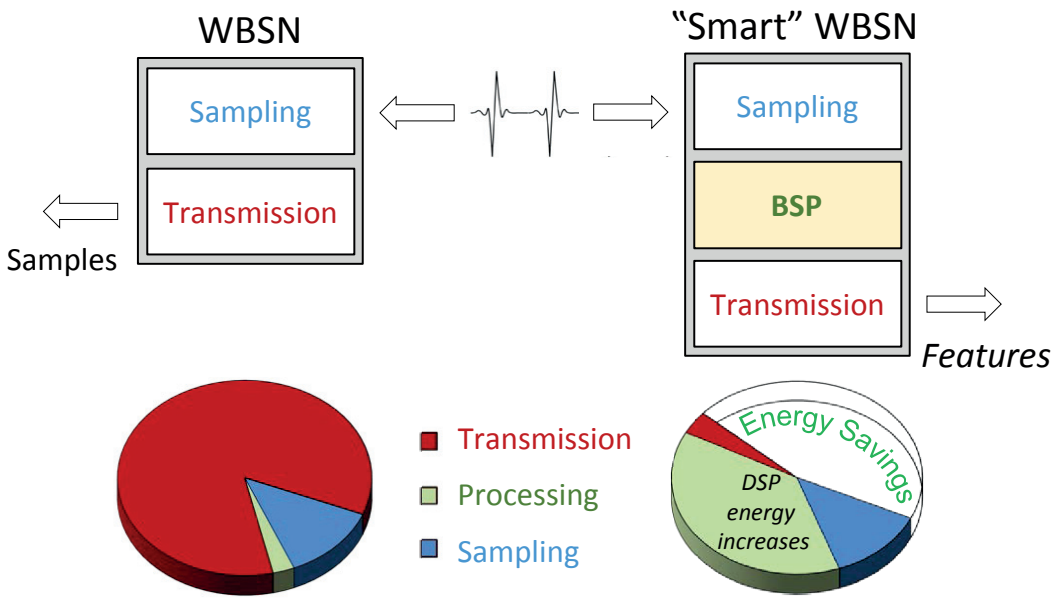


Figure 1.5 – Energy breakdown of traditional WBSNs (left) and smart WBSNs (right).

on-board DSP, it is possible to save up to 87 % of the energy spent for wireless transmission (Figure 1.5) [11]. Smart WBSNs have thereby proven to be highly useful in monitoring NCDs, while reducing the related expenses. They perform three mains tasks: (i) signal detection and acquisition, (ii) digitization and processing to extract relevant important information, and (iii) transmission of the data wirelessly through a transceiver. In this context, processing of Electrocardiogram (ECG) signals is a prime application domain for WBSNs, as they are the main indicators of cardiac diseases.

ECG signals represent the electrical activity of the heart, which is a two stage pump and its electrical activity can be measured by electrodes placed on the skin. ECG can measure the rate and rhythm of the heartbeat, as well as provide indirect evidence of blood flow to the heart muscle. Figure 1.6 features a schematic ECG for a single normal heartbeat, which shows the electrical activity on a time-scale. Normally, the ECG of a heartbeat starts with a P wave, which reflects atrial depolarization. The PR interval is the distance between the onset of P wave and the onset of the QRS complex. The flat line between the end of the P wave and the onset of the QRS complex is called the PR segment and it reflects the slow impulse conduction through the atrio-ventricular node. The QRS complex represents the depolarization of ventricles, and is a combination of the Q, R and S waves. The R wave is the most distinctive one in the ECG, and the apex of this wave is known as the R-peak. Consequently, the distance between two consecutive R peaks is called the RR-interval. The T wave represents the rapid re-polarization of contractile cells. Similarly, the distance between the onset of the QRS complex and the onset of T wave is known as QT interval while the flat line between the end of the QRS complex and the onset of T wave is called the ST segment. These are collectively known as fiducial points of an ECG.

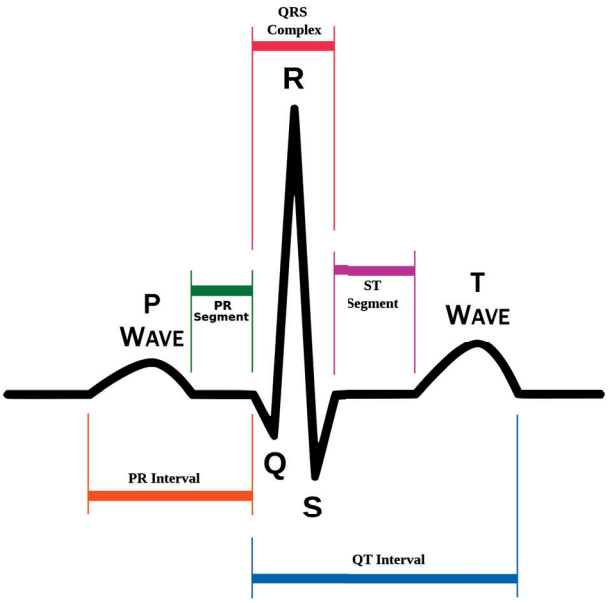


Figure 1.6 – Schematic diagram of a human ECG representing a single heartbeat.

Sensing and acquisition

The first step performed by a WBSN is bio-signal sensing, where various sensors are used to sense the corresponding signals. Commonly acquired bio-signals include ECG, EEG, blood pressure, blood oxygen saturation (SpO_2), etc. These signals received from the human body are generally blended with noise arising from sources such as breathing and muscular activity. The target bio-signal of this work is ECG, as it is one of the most widely used ones and an essential indicator of cardiac diseases. To acquire it, electrodes are placed on the human body at places like chest, arms, etc., and the difference of potential between the electrodes gives the ECG signal. The incoming ECG wave is sampled at a predefined frequency and the sensed signal is then converted from analog to digital format using Analog to Digital Converters (ADCs). The resulting digitized data is then stored in memory buffers, from which they are consequently fed to the processing cores.

Digital bio-signal processing

In this stage, the processors of the WBSNs generally execute various BSP algorithms to manipulate the raw signals. To get rid of biological noise, the signal strength is normally increased by amplification, and then it is filtered to remove noise (Figure 1.7). There are two main sources of noise in ECG signals, namely baseline wander and high-frequency muscular noise [14]. The baseline wander consists of a low-frequency component in the ECG signal (typically from 0.05 Hz to 1 Hz), that can be caused by patients' respiration, perspiration or even the misplacement of the sensing electrodes, thereby deviating the ECG from the measuring axis. On the other hand, muscular or EMG noise can add an extra high-frequency component to the

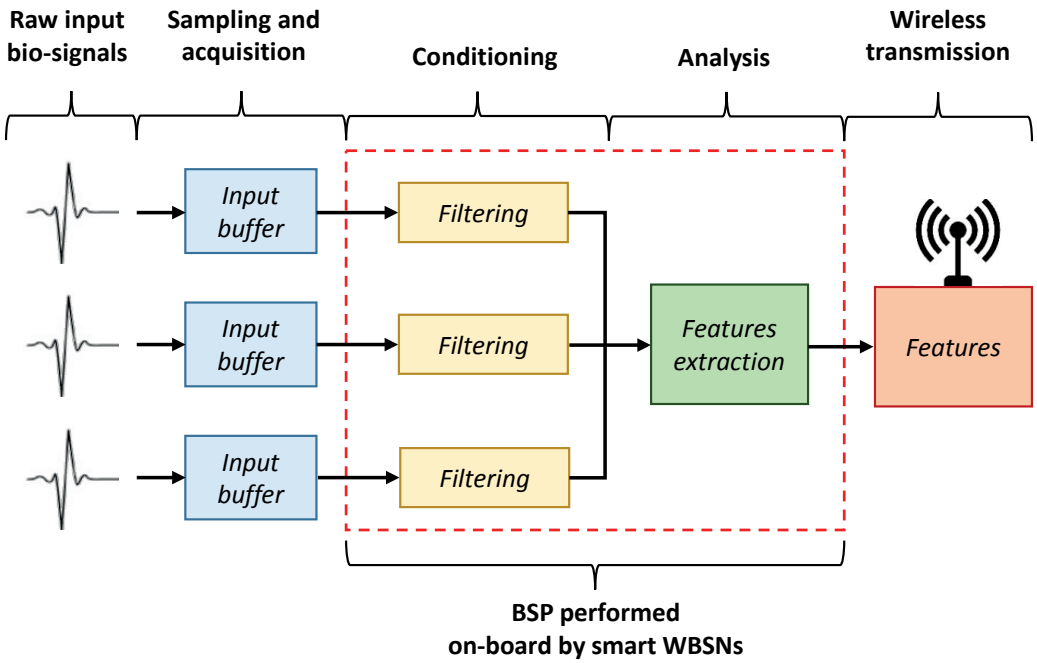


Figure 1.7 – Example of a block scheme of a smart WBSN application. In addition to signal acquisition and transmission, these devices perform BSP on-board to extract the important features.

ECG signal (over 50 Hz) and usually originates from the contraction of body muscles during movements. Various filtering techniques have been proposed to eliminate baseline wander and to get rid of external noise [14] [15]. The filtered ECG signals are then typically delineated to extract the fiducial points [16] [11]. The filtered and delineated ECG can then be used in further BSP routines such as Compressed Sensing (CS) [13], which aims at reducing the signal complexity and performs extraction of important features. The optimized outputs derived from these steps are then stored in an output buffer, from which they are transmitted over a wireless link.

Signal transmission

The extracted parameters from the processed signal are finally transmitted through a wireless link to a receiver unit. The receiver may be medical personnel, a smart-phone, a server, or a central WBSN node in charge of collecting the data from various nodes in the network, before sending it for further inspection. Prior to the advent of smart WBSNs, the transmission part used to be the most energy hungry one (Figure 1.5). Since modern WBSNs perform a wide range of BSP techniques on the node itself, and due to progresses in the design of domain-specific analog-to-digital converters [17] [18] and wireless protocols [19], the energy envelope is greatly reduced. However, as smart WBSNs are required to perform an increased amount of processing on-board, their energy bottleneck often lies in the BSP part [19] [20] [12] (Figure 1.5).

1.2 Energy bottlenecks and optimizations in WBSNs

Further improvement of energy efficiency of WBSNs therefore calls for optimizations in the BSP segment. The energy budget of the BSP segment is heavily impacted by the energy efficiency of the processing hardware as well as the software algorithms running on them. Hence, it makes *hardware/software co-design* a crucial player in this domain, calling for optimizations in WBSN designs, both in their memories and processing architectures.

1.2.1 Energy bottlenecks

This section elaborates the energy concerns of BSP for WBSNs: the memory subsystem and the processing units.

- **Memory subsystem**

  Out of the total energy consumption incurred in running BSP routines on WBSNs, the main contributor is their on-board memory subsystem (instruction and data memories) [12]. These memories mainly serve to store the instructions to be executed by the embedded processors and the input data to, or the intermediate data generated by the processors, respectively. In many cases, memories are responsible for more than half of the WBSNs' energy consumption [21]. They are also responsible for the majority of silicon real estate usage in microchips, and are forecast to have an even stronger impact in the foreseeable future (up to 95 % of the chip area in the next five years) [22], thus potentially consuming even more energy. Therefore, in order to reduce the overall energy overhead of WBSNs, it is essential to economize the energy usage of their memories.

- **Processing units**

  Traditionally WBSNs have used single-core processing architectures. However, since multi-channel acquisitions of bio-signals are often preferred for clinical accuracy, multi-core architectures for BSP have been presented, which can efficiently process task-level parallelism typical in the BSP domain. By doing so, they are able to significantly decrease the operating frequency, resulting in improvements in the energy efficiency of WBSNs (up to 66 **%** compared to single-core equivalents running at higher frequencies [23]). Moreover, supplementary hardware- and software-based optimization techniques have been proposed in the literature, which enable further energy reductions. By deploying techniques such as code synchronization, and support for Single Instruction Multiple Data (SIMD) operations, savings in the energy budget of up to 50 **%** are obtainable in multi-processor WBSNs [24]. A further opportunity for increasing the energy efficiency lies in the efficient processing of computational hot-spots or *kernels* by employing fixed function or reconfigurable accelerators. This is justified by the fact that these kernels are significantly present in BSP algorithms, as they process typically repetitive bio-signals, occupying up to 48 **%** of their total computation time [25].

1.2.2 Optimization: voltage supply scaling and management

Traditional WBSN components employ Complementary Metal-Oxide-Semiconductor (CMOS) transistor-based implementations, as they are a widely used state-of-the-art low-power option [26]. Both the dynamic and leakage power/energy consumptions of CMOS devices are tightly linked to the voltage supply (V_{dd}), with their values decreasing drastically as the supply voltage is reduced (as shown in Equations 1.1 and 1.2).

$$P_{dynamic} = n \times C \times {V_{dd}}^2 \times f \tag{1.1}$$

where:
$P_{dynamic}$ = dynamic power consumption
n = number of bits switching
C = dynamic power-dissipation capacitance
f = system clock frequency

$$P_{leakage} \propto V_{dd} \times e^{(V_{dd} - V_{th})} \tag{1.2}$$

where:
$P_{leakage}$ = leakage power consumption
V_{th} = transistor threshold voltage

This condition makes supply voltage scaling, or the reduction of the voltage that is supplied to the device, one of the most effective ways to reduce the energy consumption of WBSNs. In the past, decrease in the physical size of transistors over time, as a result of technological advances, had ensured that circuits could be operated at decreasing supply levels. This progression, known as Dennard scaling [27], derives from the fact that even as transistors get smaller, their power density stays constant. Thus, the power requirement of transistors stays in proportion with their area, thereby scaling voltage and current down with their size. This trend came to an abrupt halt around 2006, thereby making technology-enabled voltage scaling more challenging, resulting in the stagnation of supply voltages. Hence, to further reduce the power consumption of CMOS devices, the remaining option is to perform aggressive voltage scaling, implying the reduction of supply voltage for a given transistor length to levels below the nominal value. This strategy (also called *voltage over-scaling*) allows to reach the Near-Threshold Voltage (NTV) range of operation, where supply voltages approach the threshold voltage of a transistor. The rationale behind NTV operation is that transistors are most efficient when operated at, or just above, their threshold voltage (around 500 mV). While on one hand NTV operations bring about significant energy reductions, on the other hand the safe operation of WBSNs is limited by a number of factors:

- **Logic**

  The speed at which a digital logic circuit switches states (low voltage state to high voltage state and vice versa) is proportional to the voltage differential in that circuit. Reduction of the voltage supply implies a lowering of the voltage differential, thereby rendering the circuit switching slower and limiting the maximum frequency at which it can operate. This condition adversely affects the rate at which instructions can be processed by the central processing unit. As a consequence, the computation time is increased, which in certain cases can exceed the maximum permitted time-slot, thereby resulting in timing errors in some application domains.

  However, WBSN systems usually operate intrinsically at low frequencies due to exploitation of parallelism in BSP algorithms, and since bio-signals have slow dynamics. Timing errors arising out of ultra-low voltage scaling are therefore not of concern for BSP platforms. On the other hand, combinational logic circuits such as Arithmetic Logic Units (ALU) are the least affected by voltage scaling, because they are stateless and do not present internal feedback connections. Indeed, combinational circuits have been proposed operating at supply voltages in the range of few hundreds of millivolts [28]. In addition, in low-power processor architectures, their internal memories are typically implemented with register files, which are highly resilient to ultra-low voltage scaling effects [29]. Therefore, it can be concluded that the computing logic is not the resiliency bottleneck of WBSN systems.

- **Memories**

  Traditional WBSNs typically employ Static Random Access Memory (SRAM)-based memories on-board whose cells are designed with 6 CMOS transistors (6T). This choice provides a more energy-efficient solution compared to other RAM technologies, while being faster than non-volatile equivalents [30] [31]. However, SRAMs usually impose a lower limit on the supply voltage. This limit is determined by the minimum voltage level at which the memories can be reliably accessed, as, due to their construction, certain SRAM cells can suffer from different types of errors if a minimum supply voltage is not guaranteed. These errors occur when the associated Static Noise Margin (SNM) of a memory cell exceeds a given level that allows to distinguish between different logic values, thereby manifesting as a flip in the stored data bit (i.e., a stored logic '0' becomes '1' or vice-versa). SRAM errors are mainly caused by fabrication-time variations and unstable supply voltages, resulting in four main scenarios where a memory cell can potentially fail [32]:

  – *Read failures*: A data-flip caused while accessing a memory cell.

  – *Write failures*: The inability to write a certain data state into a memory cell.

  – *Access failures*: A wrong data level is read out of a memory cell due to its inability to drive the output circuitry correctly within the given access time.

  – *Hold failures*: The inability of the memory cell to retain a given data state due to a

data-flip caused by a single-event upset, such as a particle strike or coupling noise from a neighboring signal.

Opportunities and challenges

In the field of BSP, the acquired data is inherently noisy as biological signals are affected by various physical movements, such as breathing and muscular activity [33]. Thus, BSP applications typically generate outputs that are statistical or probabilistic in nature, and are inherently error-prone to varying degrees. Moreover, in these applications, the accurate processing of the signal may often be irrelevant as long as the output quality is better than the acceptable clinical threshold. For example, in the case of ECG processing, it is often enough that the heartbeat peaks and intervals between consecutive heartbeats are properly identified, rendering the rest of the signal less significant. This property permits the relaxation of reliability requirements, thereby moving from an exact computing paradigm to an inexact one. In such an approach, the accuracy of the system can be compromised to an extent to harness significant energy savings.

When WBSN supply voltages are over-scaled, thereby resulting in errors in SRAM memory subsystems, it might be possible that the corrupted outputs are still within a clinically permissible limit. Hence, voltage over-scaling-induced inexact computation enables potentially significant energy savings in WBSNs. In certain cases, however, additional memory protection schemes may be necessary (such as parity check, Error Correction Codes, ECC, or failure monitoring and recovery), in order to achieve an acceptable execution of the application. From the technology point of view, solutions have been proposed to increase the resiliency of memories against voltage scaling effects. In particular, SRAM cells with higher number of transistors (8T, 10T, etc.) have been presented, that result in lower susceptibility to ultra-low voltage scaling effects [34].

However, in both cases, either employing additional memory protection or SRAM cells with higher numbers of transistors, area and energy consumption of the memories increase substantially, thereby presenting potential trade-offs with the accuracy of the produced results by making partial and cautious use of such methods/devices [35]. Furthermore, in applications such as those measuring heart activity, the input data to be processed is generally sparse in nature, as the points of interest are focused on only small parts of the entire acquired signal. This feature not only permits the application of relaxed-reliability computation in this domain, but also of further optimization techniques such as partial protection of data.

WBSNs typically have a low sampling frequency of bio-signals (in order of a few hundred Hz) compared to processing frequencies (few Mhz typically), since the acquired signals have slow dynamics (e.g., a standard heartbeat frequency is between 1 - 1.67 Hz) [36]. The availability of sampled data to process determines the workload profile of the system. As a result, the computational trend is cyclic, with brief periods of intense processing (computation bursts), followed by intervals of minimal computation (buffering of data). This trend restricts the

processing time of WBSNs to a fraction of the total time-stamp of the acquired signal. These *idle* intervals can be taken advantage of, by applying energy optimization techniques such as powering the processing system off (power-gating) as no execution is being done. However, power-gating is not applicable to high leakage power-consuming SRAMs, as they lose the stored data upon being turned off. A traditional non-volatile alternative involving flash memories is also not viable, as real-time application requirements cannot be met due to their high write latencies [37]. A possible solution to this is shadowing of the entire SRAM data memory several hundreds of times per second to an external flash. However, the energy cost of such an approach is very high and can cancel out the potential savings obtained from power-gating, thereby bringing emerging low power-consuming non-volatile RAM memories into play.

1.2.3 Optimization: domain-specific architectures

In the literature, various strategies have been proposed for BSP architectures, of varying size, power consumption and flexibility. A standard metric for the classification of these architectures in the domain is their flexibility, which generally trades off with the energy efficiency of the respective platform. One of the most widely proposed solutions for the BSP domain is Application-Specific Integrated Circuits (ASIC) [38] [39]. Although these architectures are extremely energy-efficient, their low flexibility limits them to processing only a narrow range of applications.

On the other side of the spectrum, general purpose processing units- and Field Programmable Gate Array (FPGA)-based architectures have been proposed to increase the range of applications that can be processed [39] [40]. However, these systems are associated with high configuration latencies and hardware overhead, resulting in significant energy penalties, which is not desirable for low-power BSP applications [41]. In this context, Domain Specific Instruction-set Processors (DSIPs) have been designed for targeted application domains, such as BSP, but with a wide range of functionalities. These proposed architectures usually features a small area footprint in combination with advanced power management, performing dynamic Voltage-Frequency Scaling (VFS) to adapt to different workload conditions at runtime [29] [42] [43].

Aggressive VFS can however degrade the performance of computing platforms beyond the real-time requirements of the application. To cope with this shortcoming, multi-core architectures can be employed which support SIMD execution modes [44], exploiting the parallelism intrinsic in BSP algorithms when multiple signals are concurrently acquired and processed. Recently proposed DSIPs also feature energy-aware designs that provide the required performance at low-power operating conditions [45]. These optimized systems, featuring simplified pipelines compared to those in traditional microprocessors, have proven to be good solutions for low-power processing domains such as BSP.

Furthermore, since bio-signals are acquired from multiple channels simultaneously, the

parallel processing of the acquired data is preferred with multiple instances of DSIPs on a single WBSN device, along with the necessary hardware support [46]. On top of that, further hardware optimization techniques such as SIMD processing and code synchronization have been applied in previous works [47] [21]. In the SIMD processing scheme, all the processor cores effectively use the same instruction when they execute the same part of the application code, thereby reducing the number of accesses to the instruction memory. In the case of code synchronization, lightweight hardware support is provided to keep track of the processing flows of the different cores, in order to maximize the time when they work in parallel. This strategy boosts the SIMD operation mode, thereby bringing significant energy gains. As a result, further possibilities of energy optimization are now shifted to the processing of applications, mainly in the computation-intensive kernels that are widely present in most BSP algorithms.

Opportunities and challenges

In this context, Coarse Grain Reconfigurable Arrays (CGRAs) are a promising solution for accelerating these kernels, potentially saving computation time, and thereby increasing the energy efficiency of WBSNs [48]. CGRAs generally consist of a 2D-mesh of reconfigurable cells (also called processing elements) that are able to perform arithmetical and logical operations more efficiently than general purpose processors, and are interconnected for data-sharing. CGRAs can efficiently support the processing of computation-intensive kernels, offloading them from processors, before returning the desired outputs back to the cores for resuming their execution.

In the BSP domain, CGRAs need to work within a tight energy envelope, thereby requiring a careful design of their architectures and the selection of kernels that need to be executed by them. Moreover, as BSP algorithms typically exhibit a high level of parallelism, innovative CGRA architectures that support SIMD execution can potentially take advantage of shared control logic to further improve energy efficiency of WBSNs. In the same lines, kernels that process large input vectors in non-SIMD mode can potentially be split into subparts, thereby mapping each of them on a different set of processing elements that are governed by a shared control system.

1.3 Contributions of this thesis

WBSNs are desired to operate over long periods of time, providing continuous monitoring and analysis of bio-signals. However, since their on-board batteries have limited capacity, it is required to explore optimization techniques to reduce their energy consumption. In this thesis, I explore different approaches to increase the energy efficiency of WBSNs. My work can be broadly classified into two main parts, as follows:

1.3.1 Relaxed-reliability computation for ultra-low power bio-signal processing

In this part, I explore the application of inexactness-based relaxed-reliability computation to BSP, with the aim of trading off the accuracy of the system to harness energy savings. The prime factor responsible for energy reduction in this case is voltage over-scaling, which also results in failures in WBSN memory subsystems. As the logic circuits are far more resilient at NTV operations, especially when the system frequency is low as in the case of WBSNs, my study is therefore focused on inexactness in memories. I begin the study by exploring various *smart* memory protection schemes, where the protection is done based on the significance of the data that is stored. I further this study by doing a complete analysis of voltage over-scaling on the entire WBSN platform. In particular, first I employ the schemes applied in the preceding part, while later I waive memory protection, instead proposing specific schemes to detect errors in memories and to recover from them, thereby avoiding critical system states or failures. This study can be divided into two sub-parts:

- **Relaxed-reliability computation with significance-based memory protection**

  A majority of bio-signals consist of data that are mostly sparse, meaning that only a few portions of the signal contain useful information. In this context, I explore partial protection schemes for memories based on the significance of the data that is stored in them. This strategy is opposed to a system where the entire memory subsystem is protected by ECC techniques, that results in large energy overheads. The application of inexact computation here is directed to the data memory as BSP data is more tolerant to errors than instruction memories, where even the slightest error can irreparably damage the system's execution.

  For this study, I employ a BSP application that does the Power Spectral Analysis (PSA) of ECG signals as a case study. The PSA application outputs the ratio of power in low and high frequencies of the analyzed signal (known as LFHF ratio, which is an important clinical indicator of possible health issues [49]). This is a widely used application that employs complex signal processing techniques, such as wavelet-based Fast Fourier Transform (FFT) of the incoming ECG to do the PSA (termed as Fast Lomb method [50]). In this kind of applications, most of the data produced intermediately are typically stored in large buffers in the data memory. Specifically, the detailed contributions are:

  – A study of the statistical properties of a PSA of heart-rate variability of a given ECG to classify the data elements in the intermediate steps of the algorithm into significant or less-significant, based on their contribution to the quality of the output.
  – A bit-significance-based memory protection scheme for BSP, targeted at the non-sparse buffers.
  – A word-significance-based memory protection scheme exploiting the statistical properties of the elements produced by BSP applications, destined for the sparse buffers.

  - An exploration of the impact of voltage over-scaling, reporting the effects on the targeted data memory buffers.
  - An extension of the previous study, enforcing system-wide application of voltage over-scaling, resorting to significance-based memory protection.
  - An estimation of the effects of using voltage over-scaling with the proposed memory protection scheme on the performance of BSP applications and the energy savings that the scheme results in.

- **Relaxed-reliability computation with error management**

  The study of application of relaxed-reliability computation is further continued in this section. Contrary to the work of the previous section, herein, memory protection schemes are completely abandoned. This results in a BSP system where both the data memory and the critical instruction memory are prone to failures. The objective of this study is to quantify the inherent error resiliency of BSP applications against memory failures. In this research, I also take into account the effects of fabrication-time *variability* and supply fluctuations *(droops)* at ultra-low voltage operations, since they become increasingly prominent at low supplies. The final aim is to devise intelligent error monitoring schemes to ensure smooth recovery of processing when failures do occur. In this work, I use two popular BSP applications: one covering a completely parallel processing strategy (ECG compression), and another that exhibits a producer-consumer processing relationship (ECG filtering and compression). The main contributions of this research line are:

  - A study of the effects of voltage over-scaling on the entire WBSN platform, thereby modeling the memory failures as a function of the supply voltage, taking into account instantaneous droops as well as variabilities at fabrication time.
  - A near-exhaustive analysis of the application-visible impact of memory failures in the considered WBSN BSP routines, classifying them depending on their effect on the output.
  - A novel reliability-aware mechanism at the hardware and software levels, which guarantees system-level resilience, embodying them in an ultra-low power multi-core architecture.
  - An investigation into the efficiency of the resulting inexact system, and the ensuing trade-offs between error rates, energy consumption and output quality degradation.

1.3.2 Heterogeneous architectures for ultra-low power WBSN platforms

In this part, my research is targeted on the integration of heterogeneous architectures for BSP platforms, both at the system and memory levels. Due to the repetitive nature of bio-signals,

most of the frequently used BSP applications embody algorithms that consist of computation-intensive segments (kernels). In these lines, further optimization techniques are possible, which are mainly centered on the acceleration of such kernels on Coarse Grain Reconfigurable Architectures (CGRA) in order to harness energy benefits. More specifically, in this research line, at the system level I explore CGRAs for speeding up processing BSP applications to save energy. Furthermore, I combine heterogeneous CGRA-based WBSN platforms and relaxed-reliability computation strategies borrowed from the previous research, in order to derive energy-saving benefits from both approaches. Subsequently, at the memory-level, I introduce a heterogeneous memory architecture based on non-volatile Resistive RAMs (ReRAM). In particular, I study the energy benefits of using such a system and the endurance-related issues arising as a result. The main contributions are:

- **Reconfigurable bio-signal processing architectures**
  - An exploration of various CGRA architectures to accelerate BSP, which involves the design of the CGRA and optimized selection of CGRA parameters.
  - A methodology for kernel selection: since the CGRA has processing restrictions which depend on its design, as well as application and architectural constraints.
  - A mapping strategy for the selected kernels onto the CGRA, respecting the hardware- and software-imposed constraints.
  - A scheduling mechanism for the selected kernels, when they are executed both in SIMD and non-SIMD modes.

  The next step converges the benefits resulting out of both relaxed-reliability computation (derived from inexactness of memories under voltage over-scaling) and CGRA-based WBSN architectures.

  - A combined memory failure analysis is performed considering the diffusion of errors both in the CGRA and the processors.
  - An evaluation of the previously proposed error mitigation strategies in the current scenario, to asses their validity.
  - A trade-off analysis between the quality degradation of the output and the resulting energy savings, considering different supply voltage ranges and their fluctuations.

- **ReRAM-based heterogeneous BSP architectures**

  Shifting the focus from heterogeneous computing architectures to heterogeneous storage systems, I explore the effectiveness of integrating emerging energy-efficient memory architectures with existing WBSN systems. In particular, this work is centered on ReRAM technologies, in order to take advantage of their non-volatile characteristic and effectively power-gate the entire processing system to gain energy savings (mainly by

reducing leakage). This approach is supported by the fact that BSP applications spend an overwhelming majority of their time doing data buffering, where no processing is done. In this work, I also investigate ReRAM endurance issues, proposing lightweight measures to counter them. The resulting architecture is thus composed of a ReRAM-based main memory, while using intermediate volatile page buffers to reduce accesses to the main memory. The contributions of this part are:

- An evaluation of the important energy savings achieved with a ReRAM-based heterogeneous memory architecture for BSP applications, in comparison to SRAM-based platforms, considering various programming energy ranges.
- An endurance analysis of a heterogeneous ReRAM-based WBSN platform with a real-world BSP application, observing its effects on the device lifetime.
- A quantification of the additional memory capacity required for long-term device operation under different error-probability conditions, using a lightweight ReRAM-cell replacement strategy.
- A presentation of the trade-offs between the ensuing output quality degradation and the potential energy savings.

1.4 Thesis Outline

The rest of this thesis is organized as follows:

Chapter 2 details the proposed application of **relaxed-reliability computation in the field of bio-signal processing**, derived from voltage over-scaling. First, I study its effects on the memory subsystem and consequently on the considered BSP applications, and propose solutions to counter the ensuing errors. To do so, at the level of the WBSN memory I introduce a data significance-based protection system to preserve output quality. Then, I extend the voltage over-scaling-based inexact computation to the entire system, using the significance-based memory protection scheme to analyze trade-offs between the output quality and the achieved energy efficiency. Secondly, I apply an orthogonal approach where errors in the memories are allowed that result out of ultra-low voltage scaling in the presence of runtime droops, without any protection at the memory-level. Instead, here I propose lightweight hardware/software solutions to mitigate errors in memories by monitoring various WBSN platform components.

Chapter 3 describes the introduction of **heterogeneous architectures** (both at system and memory levels) for WBSN platforms. In particular, first, I propose reconfigurable CGRA-based WBSN architectures (conjointly with traditional processor-based ones) that are capable of efficiently processing computation-intensive BSP algorithms. This work is extended by coupling CGRA-based WBSN architectures with relaxed-reliability computation paradigm, to further decrease the energy budget, thereby studying the trade-offs with the consequent degradation of the output quality. Finally, I introduce emerging ReRAM architectures in the

domain of BSP, aimed at substantially decreasing the dominant leakage power contribution to improve the energy efficiency of WBSNs. Notably, I analyze the endurance effects on ReRAMs as a result of device-aging due to multiple writes, proposing a block-replacement solution to counter it, thereby exploring the additional ReRAM capacity required to ensure device longevity.

Chapter 4 concludes this thesis by summarizing the key contributions and observations made and by providing pointers toward future research directions in this field of work.

2 Relaxed-Reliability Computation for Ultra-Low Power Bio-Signal Processing

2.1 Introduction

Busy and unhealthy lifestyles are becoming common in today's world, resulting in a rise in the number of people developing or living with cardiovascular conditions. Moreover, a significant part of the world population is aging, and hence becoming in danger of contracting cardiac diseases. This scenario calls for increased levels of medical supervision and management, which are resulting in high costs, with traditional hospital-based healthcare infrastructures finding it increasingly difficult to cope with these demands.

In this context, emerging wireless body sensor network technologies can offer large-scale and cost-effective solutions to this issue. These wearable devices for bio-signal monitoring are bringing about a revolutionary change in healthcare systems by allowing long-term monitoring of chronic patients, while providing a low-cost and unobtrusive solution. Wireless Body Sensor Nodes (WBSN) are the building blocks of such a network, providing real-time and personalized monitoring of patients. They are designed to monitor different organs of the human body, including the heart. Such devices involve sensing of bio-signals and then transmitting them to receiver devices over wireless links, for further analysis. The current revolution in the progress of Personal Digital Assistant (PDA) technologies, like smart phones, has assisted in further boosting the use of WBSNs, thanks to the high-computational abilities of modern PDAs along with their ability to log and transmit data. However, analysis of the sensed data on the receiver side generally consists of labor-intensive manual inspection or off-line execution on a server infrastructure, calling for processing on the WBSN itself.

WBSNs can typically process a variety of bio-signals. Among them Electrocardiogram (ECG) signals, which represent the electrical activity of the heart, are clinically considered as essential indicators of the status of the heart. They are instrumental in monitoring heart activities and early detection of cardiac problems. The emergence of WBSNs has revolutionized heart monitoring as they are cheap, non-invasive and are able to acquire ECG signals continuously.

In addition, with widespread local PDA-based monitoring, medical personnel are generally alerted only in cases that require intervention. Although ECG is clinically one of the most widely used bio-signals, WBSNs are also able to monitor other activities, such as breathing, skin conductance, oxygen saturation (SpO_2) or blood pressure, to name a few. They are also used to monitor physical activities [51] [52], which finds useful applications such as in the case of monitoring patients of Parkinson's disease [53]. In such a scenario, WBSNs placed at different body-parts (chest, limbs, etc.) are used to gather kinetic information of the patients, such as acceleration and orientation.

In the recent past, a new generation of *smart* WBSNs has emerged which are able to perform Digital Signal Processing (DSP) directly on-board to analyze the acquired bio-signals (also called Bio-Signal Processing, BSP) to extract clinically-relevant features, in addition to data acquisition and transmission [54] [55]. This approach saves energy by transmitting only the relevant features instead of the whole acquired signal, however shifting the energy bottleneck of the system from the transmission part to the BSP part [30], as a result of increased on-board computation. In this work, I focus on WBSNs for ECG processing, with cardiac diseases being the major cause of deaths worldwide.

2.1.1 Energy concerns in WBSNs

In order to decrease patients' discomfort, WBSNs are desired to be light and compact devices, whose dimensions are heavily constrained. This requirement, in turn, enforces tight constraints on the size of the on-board batteries of WBSNs, restricting the total amount of energy that they can supply, and hence their time of operation. Upon exhaustion of energy, the batteries need to be recharged, which is undesirable when continuous health monitoring is required. This situation necessitates the optimization of the energy usage of WBSNs by operating them in ultra-low power conditions, so that the battery lifetime can be prolonged.

In digital systems, one of the most effective methods to achieve energy efficiency is Voltage-Frequency Scaling (VFS). This approach involves trading-off the supply voltage, and in turn the operating frequency of the system, in order to achieve energy gains. However, for systems which already operate at a very low frequency, such as ECG processing devices (order of a few MHz), voltage scaling can be successfully applied to reduce the energy consumption, while keeping the system performance intact.

In the past, traditional Complementary Metal-Oxide-Semiconductor (CMOS)-based devices (like WBSNs) have seen a trend in the decrease of their transistor sizes. Rapid advancements in fabrication technologies in the past few decades have enabled the shrinking of transistor sizes, and hence of the end-devices themselves, thus ensuring lower supply voltages. This trend, also known as Dennard scaling [27], derives from the rule that the power density in a transistor is the same for a given area for different process generations. Thus, as the dimensions of transistors were scaled down, so was the area and thereby the power density, hence bringing down required supply voltages. Dennard scaling, however, broke down around 2006, as its

rules could no longer be sustained for technologies with decreasing transistor lengths [56], thereby stalling technology-enabled voltage scaling.

This scenario led to alternative methods for energy conservation, such as aggressive voltage scaling, where supplies are scaled below the recommended threshold for safe operation. The rationale behind this approach being that both the dynamic and leakage power (and thereby the respective energy) of digital systems can be drastically reduced by lowering the supply voltage (cf. Section 1.2.2). However, such a strategy often results in the supplies approaching the Near-Threshold Voltage (NTV) ranges of transistors. Although logic circuits in embedded systems are highly resilient to ultra-low voltage scaling effects [28], the reliability of embedded memories (typically Static Random Access Memories, SRAM) decreases when approaching NTV ranges, thereby introducing failures in them [35]. However, in certain application domains that are inherently tolerant to errors (such as BSP), voltage scaling can be pushed further (referred to as voltage over-scaling), as long as system crashes can be recovered from and output degradation can be limited.

On the part of the logic, combinational circuits (such as Arithmetic Logic Units, ALU) are affected to a minimal degree by voltage scaling, as they do not present any feedback connections and do not have any operating states. Moreover, the internal memories of traditional low-power embedded processors are implemented with register files, which are also highly resilient to ultra-low voltage scaling [29]. While timing failures are typical in logic when supplies are scaled, since the operating frequency is lowered as a side-effect, nonetheless BSP architectures typically operate at low frequencies (order of a few MHz) as the acquisition of bio-signals is a slow process, thereby rendering such effects benign on the system level. Thus, as a whole, the logical part of BSP architectures does not present a resiliency challenge when supply voltages are lowered to ultra-low ranges.

Memories, on the other hand, are the main reliability concerns in BSP platforms, while also being major contributors to their total energy consumption [32] [21]. Thus, reducing their energy overheads are of primary concern for future low-power consuming WBSNs. While this task is challenging due to reduced memory reliability at ultra-low voltages, it is also rewarding as it involves saving a significant amount of energy. Traditional on-board memories for WBSN systems are designed using SRAMs as they are a power-efficient solution. However, SRAM cells are prone to failures when voltage supplies are aggressively scaled, especially when approaching NTV ranges. These failures are mainly a result of fabrication time variations that make the Static Noise Margin (SNM) of certain cells more vulnerable to scaled voltages, resulting in a flip in the stored bit (i.e., a logic '0' becomes '1' or vice-versa). Typically, the estimation of the SRAM cell failure probability involves four scenarios where the cell can potentially fail [32]:

- *Write Failure (WrF):* This type of failure occurs when a '0' cannot be written into a cell storing a '1' within the permitted time period.
- *Read Failure (RdF):* Disturbance on the bit-lines causes the value stored in the cell to flip

during a read operation. In other words, an accidental write operation happens during a read access.

- *Access Failure (AF):* The cell is not able to create a bit-differential equal to sense amplifier offset in the time it is enabled. As a result, the sense-amplifier produces an erroneous output.
- *Hold Failure (HF):* During hold mode, when reducing the standby supply voltage (V_{dd}) below a certain voltage, the content of the cell is lost.

The overall cell failure probability, P_{Total} is then derived as shown in Equation 2.1, whose value increases as the supply voltage is lowered.

$$P_{Total} = P[WrF \cup RdF \cup AF \cup HF] \tag{2.1}$$

Memory subsystems are therefore the resiliency weak links in low-power WBSNs. BSP applications such as ECG sensing, however, acquire noisy data at the inputs which is generally mixed with disturbances from overlapping biological activities such as breathing and muscle movements. These applications therefore store the noisy data in their memories, and generally produce outputs that are statistical or qualitative in nature. Therefore, errors in processed signals can be permitted in the BSP domain, as long as the generated outputs are clinically identifiable and lie within an acceptable range. To limit the detrimental effects of voltage scaling on memories, two solutions are possible: the first approach is to protect important data in the memory with additional hardware and a converse strategy is to allow errors in memory to propagate, instead relying on techniques to recover from their effects. In this chapter, I propose solutions from both of these perspectives, with memory protection schemes explored in Section 2.2 and error recovery techniques studied in Section 2.3.

2.1.2 State-of-the-art

Relaxed-reliability computation has fostered a significant amount of research, with many proposed approaches to inexact (or approximate) computation, that range from approximate hardware circuits to software-level approximations. On the software side, various methods such as loop perforation (skipping some iterations) and randomized algorithms (making random execution choices) have been proposed [57] [58] [59]. Furthermore, in domains such as multimedia processing, which require a high number of processing elements, or ALUs, it is possible to eliminate parts of circuits [60]. Such an approach, called circuit pruning, results in the loss of accuracy in the produced output, as part of the computation is ignored entirely. It mainly targets ALU sub-parts like adders or multipliers, as they are the main energy-area bottleneck of many processing systems [61]. For example, in the case of adders, their propagate and carry blocks can be simply removed. Although system accuracy is compromised, this

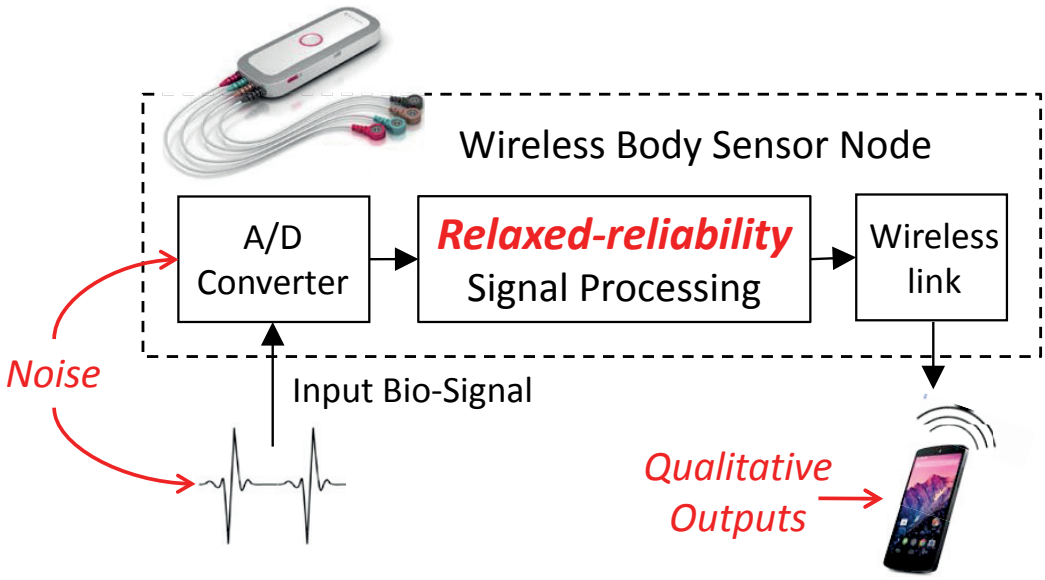


Figure 2.1 – A block scheme of a typical WBSN, highlighting the BSP segment where relaxed-reliability computation paradigm can be applied.

method ropes in large benefits by reducing the energy and area consumption. It has been reported that for error-tolerant applications, it is possible to save energy-delay-area by up to 78 %, while introducing a relative error magnitude of only 10 % [62].

Further solutions that are more fine grained also exist in the literature, including a mix of exact and inexact computation [63]. Examples of such strategies include error-tolerant adders, where the computation for the Most Significant Bits (MSB) is done using an exact implementation, while that for the Least Significant Bits (LSB) is accomplished using an inexact version. The inexact sub-adder generally uses such techniques where part of the generated output (such as carry) is completely ignored, or is set to a predefined value. Moreover, alternative methods include the creation of approximate adders, where the adder circuit is simplified (usually by employing lesser number of transistors) in order to produce outputs which are correct only for a partial set of inputs. The use of circuit pruning and other inexact adder-based computation methods are, however, not appropriate in WBSNs as their energy consumption is dominated by memories.

Another popular approach to relaxed-reliability computation is the use of inexact storage circuits [64] [65]. Instead of storing the exact data, this method involves the storage of data in memories in an inexact way, that results from static and dynamic variabilities. Ultra-low voltage scaling is one of the most prominent ways of introducing inexactness in memories. In the near-threshold range, the reliability of traditional SRAMs is particularly critical, which is compounded by voltage supply fluctuations. In addition, fabrication-time irregularities introduced in different SRAM cells lead to static variabilities. As a result, some SRAM cells are more prone to failure than others, a trend that deteriorates as supply voltages are lowered.

A further application of inexactness in memories is achieved by using bit-truncation, where the LSBs are truncated, i.e., not stored. An example of this scenario is the storage of floating-point values by truncating lower bits of the data [66]. This approach results in the loss of precision of the concerned information, and is therefore propagated to the outputs of various computational steps that use them, as well as the final output of the system. An alternative strategy is to employ memory storages rendered inexact by various methods (like lowering refresh rates of dynamic RAMs or supply voltage of SRAMs), resulting in the potential failure of memory cells [64] [32]. These failures may subsequently corrupt the data that is stored in the failure-prone cells, and hence the quality of the system's output, while promising to substantially increase its energy efficiency. Memory failures can be countered by protecting them with parity checks and Error Correcting Codes (ECC). In low-power systems, such protection mechanisms often result in high energy and area overheads. Thus, from the point of view of energy consumption, such protection schemes need to be completely or partially waived, in order to protect important data, while restricting the energy/area overheads.

The inherent error-tolerant properties of bio-signals described in the preceding section enable a paradigm shift in the computational methods that are used for their processing. In other words, it is possible to take advantage of the application of relaxed-reliability computation in the domain of BSP (as illustrated in Figure 2.1), derived from ultra-low voltage scaling. In such a scenario, the accuracy of computations may be compromised as long as clinical thresholds are respected, in order to reap potentially large energy savings.

To tackle the memory failures occurring as a result of ultra-low voltage operation, in this chapter I adopt two orthogonal strategies. First, I explore the protection of the memories themselves with additional hardware to prevent errors from spreading out of them to the processing system. To this end, I propose significance-based data protection in the memory subsystem that aims to limit the manifestation of failures in the memories into application-visible errors. In the second part of this study, I adopt a strategy without any dedicated memory-protection scheme, that allows errors to occur in memories, relying instead on hardware- and software-based system monitoring to mitigate the resulting errors. Related works in the two specific domains are discussed in details in the following.

2.1.2.1 Hardware- and technology-based memory protection

The implementation of BSP applications on ultra-low power embedded devices requires a carefully tailored energy-efficient digital architecture. Although lowering supply voltage may result in quadratic energy savings, memory subsystems using conventional 6-transistor (6T) SRAM cells become unreliable in nanometer technologies [67]. Soft errors, like bit-flips, start occurring as supplies approach NTV ranges, whose probability of occurrence increases as the supply voltage is further lowered [35]. These issues regarding reliability of memories become more prominent with modern technology scaling, as transistors with smaller lengths are conceived. For example, in [35], a non-negligible probability of error of 1.3×e-5 is reported for a V_{dd} of 0.75 V, that rapidly escalates as the supply level drops.

Since processor components are significantly more tolerant to voltage scaling [28], a possible solution to counter its adverse effects is to adopt a platform with a low voltage level for the processor and its associated registers, and a higher one for the instruction and data memories. However, such an approach requires multiple voltage regulators, as well as voltage converters on each signal crossing the boundaries of the voltage islands, thereby rendering this approach inefficient for ultra-low power platforms [63].

In this context, larger SRAM cells have been proposed in previous research work, which consist of 8-transistors (8T) or 10-transistors (10T), whose separate read and write paths ensure reliable operation of the memories at lower supply voltages compared to 6T cells [34]. However, 8T and 10T cells present a high area footprint, limiting the amount of memory that can be included in area-constrained WBSNs. The majority of the silicon real estate of typical WBSN processors is devoted to the memory subsystem, a characteristic that will become even more preeminent in the foreseeable future, as forecasts predict that as much as 95 % of the entire chip area will be devoted to memories [22]. Also, from [63] and from the International Technology Roadmap for Semiconductors (ITRS) [68], it can be seen that 8T cells occupy 30 % more real-estate, and consume on an average 25 % more dynamic energy, than comparable 6T implementations at 40 nm technology. Thus, the energy- and area-hungry 8T or 10T cells should be employed judiciously.

Another widely proposed approach in the literature to deal with errors in memories is to use ECC [69]. This method includes additional bits per memory words to detect and correct potential errors in those words. Although ECCs can potentially ensure error-free execution, nonetheless, they consume significant area and energy when all or large parts of the memories need to be protected. These characteristics make full memory protection by ECC unattractive in ultra low-power WBSNs, highlighting the need of an effective memory protection strategy. By combining different implementations of memory protection and error detection/correction mechanisms, this strategy should allow reliable operation at ultra-low supply levels, while at the same time presenting small overheads.

2.1.2.2 Hardware/software-based error mitigation

Previous works in the literature have been mainly dedicated to the application of inexact computation with error management to domains such as multimedia and Artificial Intelligence (AI), thereby making bio-signal processing a novel platform for its employment. The article [70] discusses imprecise computations related to multimedia applications and focuses on error tolerant circuits relying on their inherent capability of restricting effects of errors, possible in that application domain. However, unlike the work presented in this thesis, it does not include any error mitigation scheme. The work showcased by the authors of [71] presents an extensive review on Under-designed and Opportunistic (UnO) computing machines. In particular, it presents different efforts made to deal with hardware variability, focusing on counteracting fluctuations on the performance and power consumption due to manufacturing or aging effects. While some of the described strategies include hardware mechanisms

to detect errors of different types, the majority of them aim at mitigating the effect of such errors at the software side or by re-configuring different architectural elements (e.g., memory blocks, faulty units or even software routines). The goal of my work is instead minimizing, at the application level, the criticality of memory errors caused by transient instabilities of the voltage supply.

The work presented in [72] studies adaptive voltage over-scaling in resilient applications, however again in the graphical computing domain. The research article [73] studies the implementation of imprecise computation methods, focusing on the selective execution of various tasks based on their importance, in order to make the computation imprecise. Energy optimization techniques such as voltage scaling are not considered in their work, which on the other hand is the driving factor of my analysis. Also, unlike the system presented in my work, the analysis in [73] has been done on a platform that is not inherently erroneous upon voltage reduction, considering instead task skipping (based on the notion of computation importance) as a source of inexactness. The authors of [74] introduce an error-resilient motion estimation architecture. The errors considered in these articles are mainly timing errors, which are highly critical in such high-performance computing applications. It also analyzes voltage over-scaling and the ensuing errors, focusing on logic components. As discussed before, BSP systems operates at frequencies which are far too low to present any timing violation errors. Also, in contrast to this work, I focus on errors in the memory subsystem, since that is the main susceptible element in WBSNs at ultra-low voltages, whereas the processing logic remains largely unaffected. In fact, in [72] the authors have used Razor based registers [75], which can detect timing violation errors that are not of concern for BSP applications as they are typically processed at very low frequencies. Although these errors are detected using a lightweight scheme, recovery from them is not performed in their work. Instead, it uses an adaptive voltage supply that can be increased when the error counters cross a certain threshold so that no further error occurs, and then lowered again after a certain amount of time has elapsed. Also, in comparison my work also details how recovery is done in case of a system crash, which is not reported in [72].

The article in [76] discusses inexact computing in multimedia and AI applications. It analyzes error tolerance at the application level, and specifically discusses the effects of transient or soft errors. The considered errors have been introduced in various hardware components like the issue queue and the physical register file. My work fundamentally differs from this in a number of ways. First, while this work focuses on soft errors in certain hardware storage components, in my work I have considered errors for the entire simulation length and in the entire memory footprint considered. This approach makes the proposed system cover both transient errors and hard errors in the memory subsystem, thereby providing a more extensive analysis. Secondly, the error recovery technique reported in this work relies on a check-pointing strategy to recover from critical errors. For errors causing a system hang, they use a watchdog timer for alerting. I instead employ a different approach relying on lightweight checks for the correctness of task-synchronization operations, memory accesses and FIFO surveillance on the input Analog to Digital Converters (ADC).

2.1.3 Contributions and outline of the chapter

In this chapter, I apply ultra-low voltage over-scaling as an energy-saving strategy in WBSNs, proposing solutions to counter their adverse effects on memory reliability. The main contributions of this chapter are:

Relaxed-reliability computation with significance-based memory protection

- I perform a study of the statistical properties of the data stored, considering the Power Spectral Analysis (PSA) of ECG as a target application. The stored data is classified into sparse and non-sparse buffers.
- I propose a data bit-significance-based memory protection scheme for non-sparse data buffers, partially protected using ECC.
- I propose a word-significance-based memory protection scheme for sparse data buffers, with partial ECC and partly parity check-based protection. The corrupted data is replaced with the statistically expected value, rather than doing correction.
- I explore the trade-offs possible by varying the degree of protection of memories, thereby the output quality, and the energy savings obtained.
- I extend this research by applying aggressive voltage scaling to the entire processing architecture, including the memory subsystem. As a result, I investigate the trade-offs between the quality of output and the energy efficiency.

Relaxed-reliability computation with error management

- I model the effects of ultra-low voltage scaling on WBSN SRAM-based memories, considering the effects of voltage supply droops and fabrication-time variabilities of SRAM cells.
- I perform an in-depth analysis of the impact of memory failures on a set of BSP applications that are run on the system in order to classify them depending on the criticality of application outputs.
- I propose hardware/software-based error recovery mechanisms, that are able to recover the system from crashes, while minimizing effects of memory failures.
- I study the trade-offs between the degradation of application outputs as a result of voltage over-scaling, and the resulting energy benefits, thereby identifying an optimal point of operation.

The remainder of the chapter proceeds as follows. First, I introduce a heterogeneous data significance-based memory protection scheme for WBSNs to counter the adverse effects of

voltage supply scaling that is used to improve their energy efficiency. While in the Section 2.2.2, I focus on the data memory subsystem only, in the following one (Section 2.2.3), I extend these techniques to the entire WBSN processing platform, studying the trade-offs that can be achieved in the energy savings by compromising the output quality. Then, in Section 2.3, I explore the application of relaxed-reliability computation with error recovery as an energy-saving strategy in the field of BSP, discussing the challenges faced and proposing solutions to counter them. Finally, Section 2.4 summarizes the research contributions made and the results obtained in this chapter.

2.2 Relaxed-reliability computation with significance-based memory protection

2.2.1 Introduction

Voltage over-scaling is a widely employed method for energy reduction in embedded systems (like WBSNs) as it gives quadratic reductions in the energy budget (cf. Section 1.2.2). However, operations at ultra-low voltages makes WBSN memories prone to errors, thereby threatening the correct execution of Bio-Signal Processing (BSP) applications. However, BSP algorithms typically processes noisy inputs, generating *sparse* intermediate data, before producing statistical outputs. Thus, they are inherently tolerant to a degree of inexactness introduced by ensuing bit-flips in memories as a result of ultra-low voltage operation. Energy hungry full protection of the memory subsystem is therefore not a necessity as long as an acceptable output quality can be assured. This condition permits to take advantage of the sparsity of the data stored in the memories to relax the degree of protection for them, since such data have potentially limited impact on the output accuracy. In addition, due to the typically low operating frequency in the BSP domain and resilient logic elements, errors in the logic part are mostly negligible, therefore calling for smart energy-efficient solutions to protect the memory subsystem.

In this section, I explore the application of relaxed-reliability computation in WBSNs derived from voltage over-scaling. To deal with the ensuing errors, herein I propose a significance-based protection scheme for the data memory. It uses ECC for protecting words that are not sparse, while just detecting an error when the statistical prediction is available for predominantly sparse memory sections, replacing them with the expected value. I begin this research by applying the proposed approach only on buffers present in the data memory, as it constitutes a significant energy bottleneck in WBSNs. Consequently, I perform the study of applying voltage over-scaling to the entire WBSN processing system, adopting the significance-based strategy for data memory buffer protection. This is done in conjunction with full protection for the instruction memory and the rest of the data memory. The approaches are then evaluated against a real-world BSP application that does the power spectral analysis of ECG (which is a powerful tool for predicting heart problems [49]).

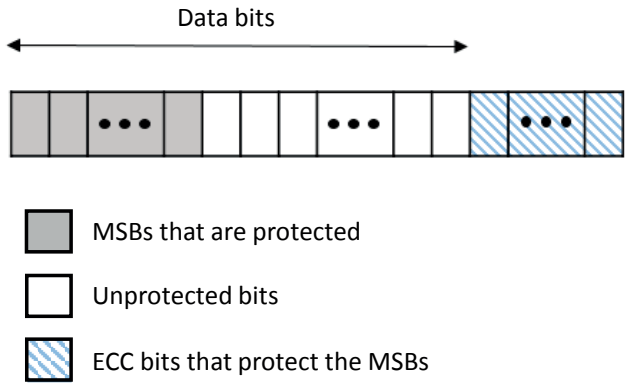


Figure 2.2 – ECC-based protection for non-sparse buffers.

2.2.2 Ultra-low voltage scaling in the data memory

In this part, I apply voltage over-scaling to the data memory, proposing a novel memory protection scheme that takes advantage of the *sparsity* inherent in acquired bio-signals. At the same time, it utilizes the benefits of memory protection schemes involving additional bits by partially employing them. To do so, I first identify the data distribution in the data memory subsystem while executing a commonly used BSP application, classifying them into *sparse* (i.e., with magnitude close to zero) and *non-sparse* (i.e., with magnitude far from zero) segments (termed as buffers). Then, I apply two heterogeneous protection schemes for the memory buffers storing the two types of classified data, the detailed implementation of which are described in the following.

2.2.2.1 Data-bit significance-based protection for non-sparse buffers

Different data distribution patterns in BSP applications indicate that distinct protection approaches against memory faults can be applied for limiting the incurred overhead. As each element in the non-sparse buffer is potentially significant to the output calculation, all of them need to be protected, even if partially. In such a data distribution, the MSBs generally store information that can impact the computation results the most. For taking advantage of the error resiliency of such a BSP application, a protection scheme can be applied in which the MSBs of every word in the *Extr_buffer* are protected with a State-of-the-Art (SoA) mechanism such as ECC. The LSBs are not protected against memory faults by any specific mechanism, as depicted in Figure 2.2, therefore making the computation results susceptible to bit-flips in them. By employing m ECC bits, it is possible to protect 2^m*-1* bits, which indicates 2^m*-m-1* data bits when the ECC bits are subtracted [77]. A complete list is shown in Table 2.1. In this case, I have considered Single Error Correction and Double Error Detection (SECDED) ECC for implementing MSB protection.

Table 2.1 – Number of SECDED ECC bits required to protect data bits.

ECC bits	Total bits	Data bits
2	3	1
3	7	4
4	15	11
5	31	26
6	63	57
7	127	120
8	255	247

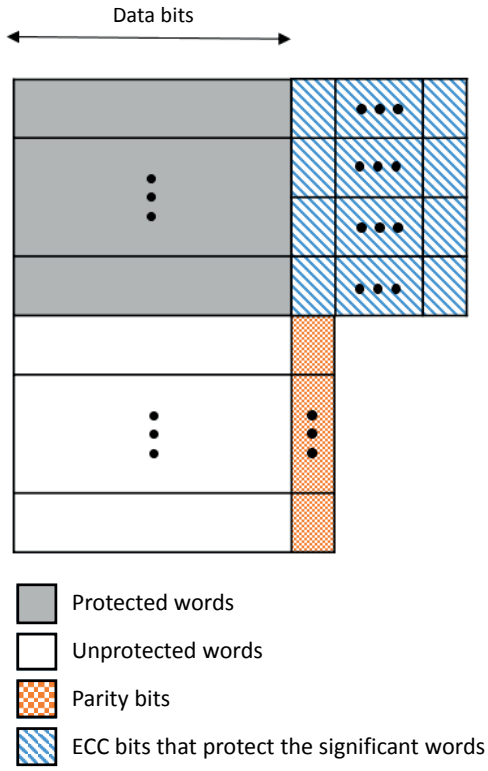


Figure 2.3 – ECC- and parity-based protection for sparse buffers.

2.2.2.2 Word significance-based protection for sparse buffers

On the other hand, in case of the sparse buffers, the distribution of the stored data allows the application of a more relaxed protection scheme. As opposed to non-sparse buffers, here the discrimination is made between significant and non-significant *words*. In particular, rather than protecting groups of bits, complete words can be protected, distinguishing between more- and less-significant ones. As a majority of the words are close-to-zero, it is possible to replace them with their expected value (zero) if a bit-flip occurs in a word. This strategy ensures that the impact of such an error will not drastically affect the expected data, since a

flipped bit within each of the close-to-zero data can alter completely the magnitude of the stored value. To implement this strategy, just error-detection supported by a single parity bit can be employed per less significant word, thereby resulting in a small overhead with respect to error correction. For the significant data, which has magnitudes much larger than zero, a more expensive SECDED ECC scheme can be applied for ensuring their correct storage. The resulting heterogeneous scheme is depicted in Figure 2.3.

2.2.2.3 Experimental setup and results

This section describes the results obtained by the application of the proposed heterogeneous and relaxed memory protection scheme on a target BSP application that performs the power spectral analysis of ECG. In this study, the voltage scaling effects are considered only in the intermediate memory buffers, while a complete version is presented in Section 2.2.3. Herein, I describe the experimental setup that was used to evaluate the strategy, varying the levels of protection in the two considered memory buffers. This study enables the investigation of the trade-offs between the degradation of output quality and the energy efficiency that is obtained as a result of varying the degree of memory protection.

Case study: power spectral analysis of ECG

Many algorithms have been proposed in the literature to predict heart diseases, ranging from the automated detection of epileptic seizures [78], to the predictive risk assessment of atrial fibrillations [79]. Out of these, Power Spectral Analysis (PSA) of ECG signals is a powerful tool for evaluating the autonomic control of the heart rate and for identifying various health conditions [80] [81]. This BSP application analyzes the heart-rate variability in order to derive the power in the high- and low-frequency regions of the signal. In clinical practice, the most used metric obtained from PSA is the ratio between the power in low frequencies (LFP, defined as 0.04 Hz to 0.15 Hz) and high frequencies (HFP, 0.15 Hz to 0.4 Hz), with LFHF ratio = LFP/HFP. A deviation of the LFHF ratio above or below normal values is indicative of various health issues [49].

Various methods can be used for the estimation of the periodogram, which is the output of a PSA system; but due to the non-periodic nature of the processed RR-intervals, the Lomb method is one of the most suitable ones [82]. The Lomb method is relatively complex, and further research has led to the creation of a more optimized Fast-Lomb method [50]. This approach uses two complex Fast-Fourier Transforms (FFT) for reducing the sums over trigonometric functions in Equation 2.2 to four less-complex sums, obtaining the Lomb periodogram of a non-periodic signal (x_j):

$$P_N(\omega) = 1/2\sigma^2.\left\{\frac{[\sum(x_j-\mu)cos\omega(t_j-\tau)]^2}{\sum cos^2\omega(t_j-\tau)^2} + \frac{[\sum(x_j-\mu)sin\omega(t_j-\tau)]^2}{\sum cos^2\omega(t_j-\tau)^2}\right\} \tag{2.2}$$

where μ and σ are the mean and the variance of the signal, respectively. τ is a constant offset for each angular frequency ω that makes the periodogram invariant to time-shifts, while t_j is the time interval of the RR sample (cf. Section 1.1.2), and the sums are taken over the corresponding window size [83].

The PSA system has four main steps, described as follows.

- **Extraction of RR intervals:** In the first step, the time differences between consecutive heartbeats (known as RR intervals) are extracted from ECG recordings of patients. The RR intervals are non-periodic signals, as the periods between consecutive heartbeats are not constant. Therefore, they require processing by dedicated algorithms, such as the Fast-Lomb method, to create an evenly spread periodogram.
- **Extrapolation:** In the next step, according to the Fast-Lomb method, the extracted RR intervals are extrapolated to a fixed size window (e.g., 512 samples to match the fixed size of the FFT). This procedure essentially converts the non-periodic signals into uniformly sampled ones. The FFTs (implemented with split-radix method, which is one of the fastest known FFT realizations) then calculate the four sums in Equation 2.2 for each window and the Lomb calculator combines the data in order to provide the real-time power-spectrum information.
- **Wavelet-based FFT and Lomb calculation:** The uniformly-sampled data are then processed to estimate the specific trigonometric functions required by Fast-Lomb. Traditionally, such an estimation is performed by applying FFT. Instead, and similarly to [50], in this work a Wavelet-based FFT (WFFT) is employed, which reduces substantially (up to 28 % compared to the SoA) the complexity of the Fast-Lomb method [82], while also introducing sparsity in the data being processed [84]. In particular, the wavelet transform involved in the WFFT helps in revealing the sparse nature of bio-signals in the wavelet domain, eventually exposing the terms that are zero (or close-to-zero). Such close-to-zero terms and the following butterfly operations applied in the second stage of the WFFT can then be pruned, consequently reducing the computational complexity. Finally, the Lomb calculator combines the output data obtained from WFFT, estimating the real-time power spectrum information.

Analysis of the PSA system

Figure 2.4 shows the schematic diagram of the PSA application flow. The target system first performs the extrapolation of the input RR-intervals and stores the data in a memory buffer, named *Extr_buffer*. A wavelet-based FFT is then performed on the uniformly-sampled data in order to estimate the specific trigonometric functions required by the Fast-Lomb algorithm, which also introduces sparsity in the processed data. The output of this step is stored in a memory buffer called the *DWT_buffer*. Butterfly operations are performed on the data in the *DWT_buffer* to generate the periodogram of the input ECG [82]. Finally, the Lomb

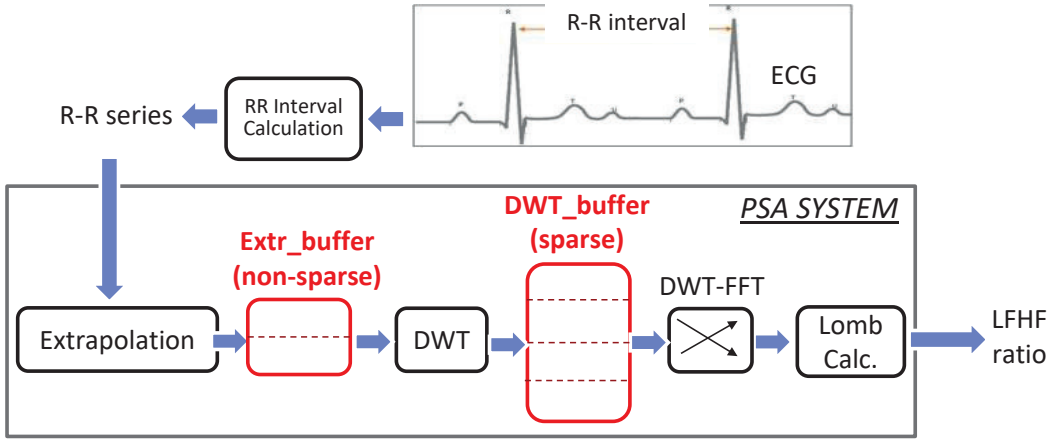


Figure 2.4 – Block-scheme of the PSA system.

calculator analyzes the resulting periodogram in order to extract the power in the low- and high- frequency ranges. By doing so, it provides the most clinically important data obtained from the PSA of ECG signals, i.e., the LFHF ratio.

The application of an efficient memory-protection scheme requires the identification of the statistical characteristics of the target application for determining blocks of data where it can be employed. In order to do that, the PSA system was analyzed and the data distribution in its various stages were estimated by performing several experiments with ECG recordings.

Figure 2.5 showcases the trend of the data that is stored in the *Extr_buffer* and in the *DWT_buffer*. The elements of the *DWT_buffer* (Figure 2.5.b) are mostly centered on zero, thus having a sparse nature, while the elements of the *Extr_buffer* (Figure 2.5.a) have a non-sparse distribution. This implies that the more important part of the data in *Extr_buffer* is stored in the MSBs, while the LSBs have less impact on the value of the data. Thus, a failure in an MSB will potentially have an impact on the computation that is much more adverse compared to a bit-flip in an LSB.

For the *DWT_buffer*, however, the bit positions are important only to the data that are not centered-on-zero, and as in the *Extr_buffer* an MSB flip is more likely to have a negative impact than an LSB flip. Since the expected statistical value of most of the stored data is zero, the significance is on the granularity of words rather than bits in these data.

A safe approach toward protecting less significant words in case of a bit-flip in it is to replace the word with zero, which is the *expected* value for it. By employing a simple parity-based check, this can be achieved by replacing an affected word with a value (zero) that has a small relative degradation. In this way, the adverse effect of bit-flips in changing the value of the data by orders of magnitude is avoided, although an error of a potentially small magnitude is introduced.

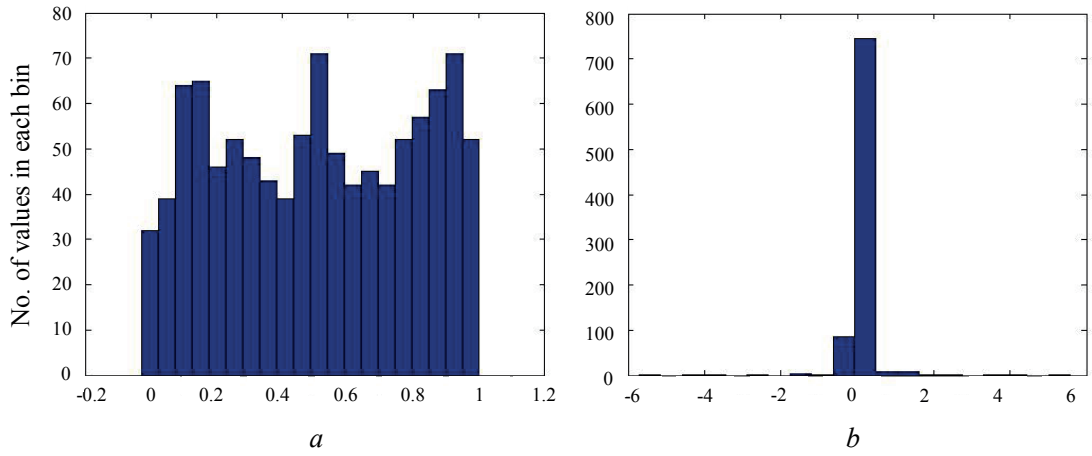


Figure 2.5 – Statistical distribution of the data in the different buffers: a) *Extr_buffer* and b) *DWT_buffer*.

- **Bit-based protection for *Extr_buffer***

  For the non-sparse *Extr_buffer*, a bit-significance-based memory protection strategy is adopted (cf. Section 2.2.2.1). In this case, six SECDED ECC bits are required to entirely protect a 32-bit word (cf. Table 2.1), as used in the data memories of the target architecture (cf. Section 2.2.2.3). A lower number of ECC bits may be employed when partial protection is desired.

- **Word-based protection for *DWT_buffer***

  On the other hand, in case of the sparse *DWT_buffer*, a word-significance-based memory protection approach is employed (cf. Section 2.2.2.2). In the PSA application, sparse elements typically reside in the low-frequency outputs of the DWT, making a parity-bit-based error detection sufficient. For the more significant words, a full SECDED ECC-based protection is needed to ensure error-free data in them.

Framework

To evaluate the proposed hybrid memory protection scheme, the input ECG data was retrieved from the PAF prediction challenge database, available on the Physionet portal [85]. The database includes 100 recordings, of 30 min each. I have considered input data windows of 2, 4 and 6 minutes, with an overlap of 1, 3 and 5 minutes, respectively, which provides a realistic time-stamp for analysis. The data from each window in each recording were independently processed by the application in real-time to retrieve their LFHF ratio.

For mimicking bit failures, I have adopted an approach based on the generation of random error masks. In this scenario, masks are generated with their size corresponding to that of the unprotected part of the memory buffers. They are a 2D binary array whose number of rows equal the word size of the memory buffers (32 bits) and whose number of columns represent

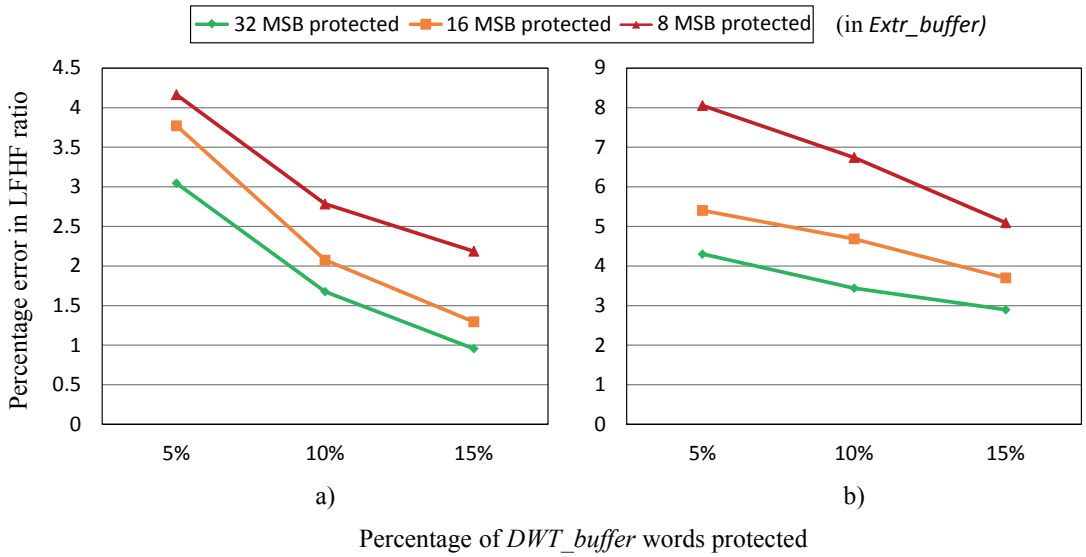


Figure 2.6 – Percentage of error in the calculation of the LFHF ratio under the different protection schemes: a) at 650 mV and b) at 600 mV.

the number of words stored in the corresponding memory buffer. In an error-free case (a '1' in a mask cell represents a bit-flip) all bits in the array are '0'. However, at low-supply voltages, errors are randomly inserted in the matrix with a probability of occurrence corresponding to that supply voltage [35]. Different error masks are employed for each processed input window and for each buffer, but the same set is used across all protection configurations (explained in the following). For all the buffers, data is represented with 32-bit words.

For the *Extr_buffer*, I explored the protection of 8, 16, or 32 (all) MSBs, while for *DWT_buffer*, I assumed a protection of 5 %, 10 % or 15 % of the most significant memory words. A simple sorting algorithm was employed in the case of *DWT_buffer* to separate the non-sparse data from the sparse data, placing the non-sparse data on the top of the memory buffer, while the rest below it, in a decreasing order of magnitude. To evaluate the proposed heterogeneous protection scheme, a high-level fault simulation environment was developed, executing the entire target application. In this way, I evaluated the impact of errors in the intermediate buffers on the quality of the PSA output, which was compared under different protection schemes to a fault-free execution. Single bit-flip errors in the buffers were considered with probabilities of 0.07 % and 0.22 %, corresponding to the behavior of a 6T SRAM at 650 mV and 600 mV, respectively [35], which fall in the ultra-low power NTV ranges.

Error analysis

Results from each recording and each window size are averaged in the presented results. Figures 2.6.a and 2.6.b compare the percentage error in the computation of the LFHF ratio at supply voltages of 650 mV and 600 mV, respectively. The obtained results highlight that the

selective protection of significant words in sparse buffers can still ensure low degradation in the output quality, with little additional overhead. In fact, in the case of a voltage supply of 650 mV (Figure 2.6.a), less than 1 % error in the LFHF ratio can be achieved by protecting only the most significant 15 % of the words in the *DWT_buffer*, when the non-sparse *Extr_buffer* is completely protected. As expected, by reducing the ratio between significant and non-significant words in *DWT_buffer*, thereby protecting fewer significant words, the error introduced in the LFHF ratio increases. Nevertheless, the introduced error still remains rather low at 3 % even when only 5 % of the most significant words are protected. The percentage error nearly doubles and triples when going from 15 % protection of the significant words to 10 % and 5 %, respectively, when the *Extr_buffer* is error-free. Similar trends are also observed when 16 and 8 bits are protected in the *Extr_buffer*, but with higher error magnitudes, respectively, which is expected as more bit-flips occur in the *Extr_buffer*.

Conversely, the protection of the non-sparse *Extr_buffer* presents more challenges, as all the data stored there are potentially critical. Even when protection against bit-flips is provided for the 16 MSBs of each word (which corresponds to protecting half of the buffer content), a noticeable decrease in quality-of-service can be noted (square-dotted orange line in Figure 2.6.a). The same trends can be noticed in Figure 2.6.b for a lower voltage supply of 600 mV. In this case, the relative error has a greater value as the number of bit-flips is also much higher. Interestingly, even in this scenario, the deviation in the LFHF ratio, with respect to a fault-free execution, can be restricted to 5 % by allowing errors in the 16 LSBs of *Extr_buffer* and only checking (but not correcting) errors in 90 % of *DWT_buffer*.

Energy analysis

The energy overhead induced by the memory protection configurations has been comparatively evaluated by modeling the different schemes using CACTI [86] in the McPAT framework [87]. The *Extr_buffer* requires two buffers, each of 8 kB, while the *DWT_buffer* is composed of four buffers of 8 kB each. The operating temperature has been assumed to be 300 K. The technology node employed in the simulation of the memories is 40 nm. All wirings were considered to be global, and the interconnect projection was taken as conservative. The memories have a single port used for both reading and writing. Also, in my experiments, I have assumed that each of the memories consists of a single bank connected using a bus.

The CACTI model outputs relevant metrics, consisting of the dynamic read and write energies per access and the leakage power of the target memory configurations. To derive the corresponding dynamic energy, the number of accesses to the different buffers from the high-level model of the application were retrieved by employing software counters. In addition, the leakage energy was estimated by considering the execution time of an optimized version of PSA running on an ARM Cortex M4 processor operating at 180 MHz [88]. Moreover, additional storage is required to support data protection. In the case of *Extr_buffer* (protection of most significant bits), considering one error detection and correction per memory word,

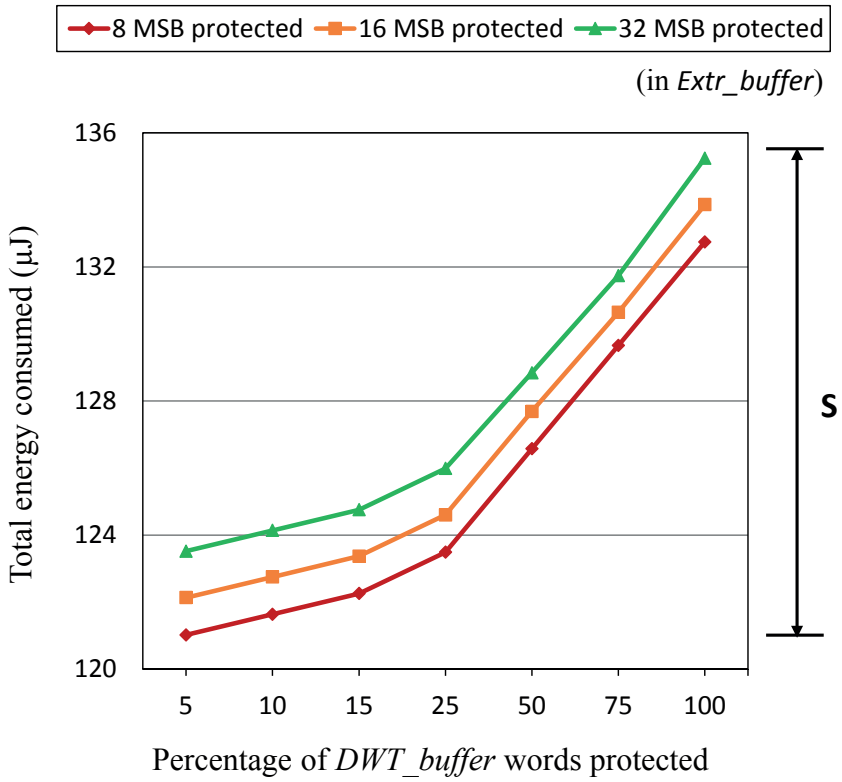


Figure 2.7 – Total energy consumption at 650 mV supply under the different memory protection schemes for an execution time ≈ 2.67s.

6 extra ECC bits are required for each word when all 32 data bits are protected. Fewer ECC bits are used when only the most significant part of each word is protected: 5 and 4 bits in the case of 16- and 8-MSB protection, respectively. A maximum of one error occurring per memory word has been assumed and the number of ECC bits were chosen accordingly as described in Section 2.2.2.3.

In the case of the *DWT_buffer* (protection of most significant words), a single parity bit is employed for less-significant words, allowing error detection. Single error correction is instead supported for significant words by using 6-bit SECDED ECC. To simulate this heterogeneous structure in CACTI, I have considered two corner cases where all data is protected either employing 1-bit parity check or 6-bit ECC protection. To derive the dynamic energy per read access of intermediate configurations, I have employed Equation 2.3:

$$E_t = p \times E_p + (1-p) \times E_u \tag{2.3}$$

where E_p is the read energy per access in the protected part of the memory, and E_u is the read energy per access in the unprotected memory. Also, p is the percentage of the considered

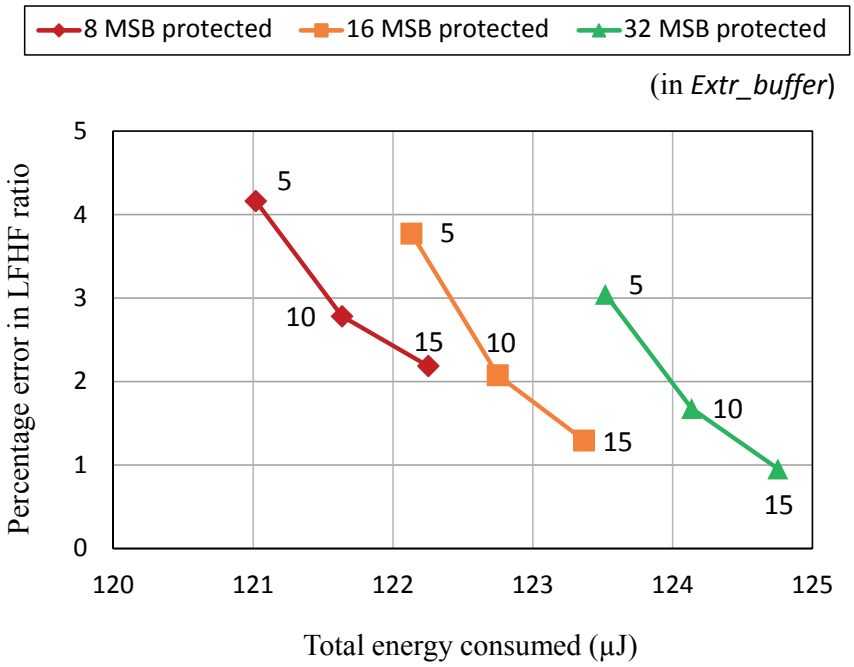


Figure 2.8 – Percentage error in the LFHF ratio and the corresponding energy consumption under the different studied memory protection schemes. Numbers beside the points represent the percentage of *DWT_buffer* protected.

significant words, whereas E_t is the net read energy of the heterogeneous memory structure. The write energy per access and the leakage power of the hybrid memory were also calculated in the same manner. For this study, I have considered a worst-case scenario that corresponds to the maximum number of accesses to the hybrid memories for running the PSA application using the RR intervals from a patient recording of 5 min. To obtain the leakage energy from leakage power figures, I have measured the runtime of the PSA application on all input windows on the target ARM Cortex-M4 processor, which corresponds to 2.67 s.

Figure 2.7 details the energy consumption of the hybrid memory system under the different protection schemes employed. The variable '*S*' represents the energy saved when 8 MSBs are protected in the *Extr_buffer* and 5 % of the significant words are protected in the *DWT_buffer* (baseline configuration), with respect to full protection (32 MSBs and 100 % of the words). It shows that full protection of all the data in the memory buffers entails a significant energy overhead of 12 % with respect to the baseline configuration. On the other hand, by having a 16-bit MSB protection in the *Extr_buffer* and 10 % of significant words protected in the *DWT_buffer*, only 1.4 % of energy overhead is acquired. As reported in Figure 2.7, this configuration induces a negligible LFHF ratio degradation of just 2 % at 650 mV, which mounts to 4.7 % at 600 mV. Of course, with a better protection scheme like 32 bits in the *Extr_buffer* and 15 % of significant words of the *DWT_buffer*, the error in the LFHF ratio is lowered to 0.96 % at 650 mV, but the energy overhead over the baseline is increased to about 2.5 %.

Energy/quality-of-service trade-offs

The proposed methodology can be employed to determine the optimum memory configuration when a constraint on the output error is given for the target BSP application. Alternatively, it can also be used to devise the most appropriate memory protection solution for a given energy budget. Figure 2.8 illustrates this trade-off between the total energy consumed by the buffers in the PSA application and the percentage of error in the computed LFHF ratio. First, I study the case when a percentage of bit-flips of 0.07 % is considered, corresponding to a supply voltage of 650 mV. For an assumed maximum tolerable error of 3 %, a solution with 8 MSBs protected in the *Extr_buffer* and 10 % of the significant words protected in the *DWT_buffer* is the most energy-efficient. Conversely, if an energy budget of 123 µJ is specified, the smallest error in the output is achieved by protecting just 10 % of the significant words and 16 MSBs in the *DWT_buffer* and *Extr_buffer*, respectively. Hence, using this research, it is possible to determine the best memory configuration suited to the needs of the application.

2.2.3 Extension of ultra-low voltage operation to the entire WBSN system

The work in Section 2.2.2 studies the impact of relaxed-reliability computation on memories, focusing on the effects of voltage over-scaling on the intermediate memory buffers. It has been shown that considerable energy benefits can be obtained with low introduced error due to ultra-low voltage scaling in data memories, by judiciously employing significance-based memory protection schemes. Thus, it forms the basis for further exploration of the effects of application of supply voltage over-scaling on the entire WBSN BSP architecture, including the whole memory subsystem. In this research, memory protection techniques similar to those in Sections 2.2.2.1 and 2.2.2.2 are employed for intermediate buffers in the data memory. The objective again is to take advantage of the sparse intermediate data to incorporate a word-significance-based memory protection system in the corresponding buffers, along with a bit-significance-based protection for non-sparse data. This study shows how the previously proposed memory buffer protection schemes fare on the system level and allows the evaluation of their impact on the output/energy consumption, when voltage over-scaling is applied to the entire BSP system.

I have applied the proposed methodology to a typical WBSN BSP architecture (depicted in Figure 2.9), which typically comprises a low-power processor along with supporting memories. The digitized input ECG is typically stored in the Data Memory (DM), which is later used by the processing core during execution. It has been assumed that the application instruction content is transferred into Instruction Memory (IM) during bootstrap phase. This initialization phase only requires a tiny fraction of the total runtime of typical WBSN acquisitions, which can last from hours to days. My research therefore focuses on the architectural components active at runtime: the microprocessor and the memory subsystem (Figure 2.9). Details of the conducted experiments are as described in the following sections.

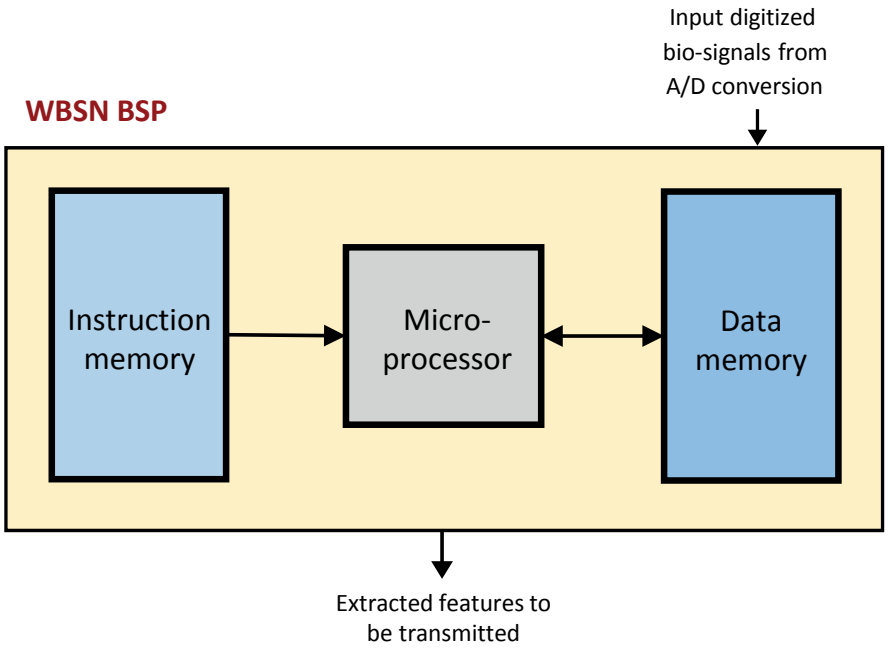


Figure 2.9 – Block scheme of a typical WBSN's processing unit, representing the components active during bio-signal processing.

2.2.3.1 Experimental setup and results

This section first details the experimental setup that was adopted to validate the proposed strategy on the entire WBSN system. Following that, it discusses the obtained results by performing an analysis of the error in the system output under different protection schemes, the resulting improvement in the energy efficiency, and the possible trade-offs between them.

Framework

The data memory, realized with 6T SRAMs, is itself divided into two sections (Figure 2.10). The first one comprises the part outside the *Extr_buffer* and *DWT_buffer*, which must operate without errors irrespective of the supply voltage. It is therefore entirely protected by SECDED ECC codes. Similar to Section 2.2.2.3, for the second part of the DM section that is composed of the buffers (*Extr_buffer* and *DWT_buffer*), abbreviated as *DM_Buff*, I have considered a maximum of one error occurring in a memory word. For the non-sparse *Extr_buffer*, I have also used SECDED ECC-based protection for the memory words. Three cases were investigated: 32 bits (full word) protection requiring 6 ECC bits, 26 MSBs protection, the maximum that can be achieved by 5 ECC bits, and 11 MSBs protection that requires 4 ECC bits (cf. Table 2.1). In the case of the sparse *DWT_buffer*, I have employed 6 ECC bits for the protection of the most significant words, and one parity bit for detecting errors in the less significant ones. The IM always needs to operate reliably, as it stores the instructions for the application execution,

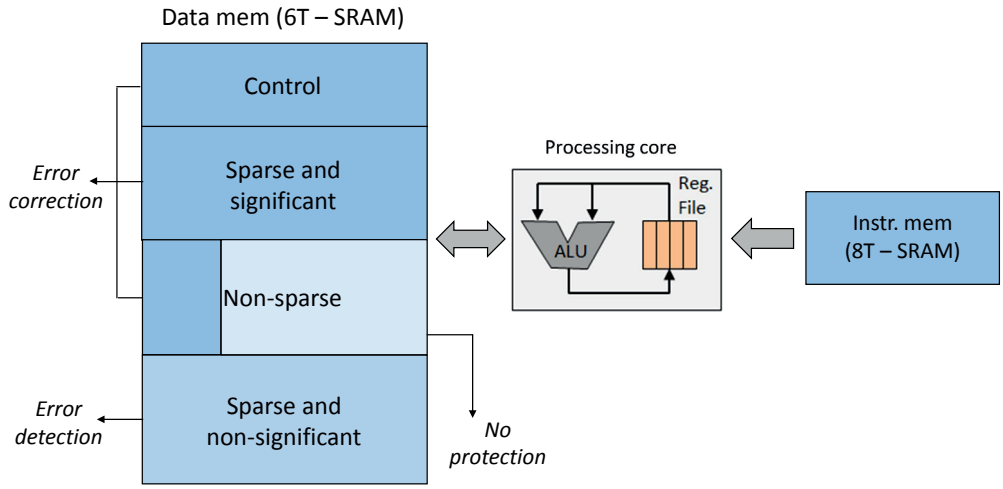


Figure 2.10 – The proposed heterogeneous protection scheme applies varying exactness guarantees depending on the data criticality.

where any bit-flip in it can potentially result in an undesirable system crash. It is implemented by using error-resilient 8T SRAM cells. The static power and dynamic energy figures of the IM were obtained from CACTI 6.0 memory modeling tool [89], adapting them to the target 40 nm technology. The IM has a size that is able to store the whole application.

Similar to Section 2.2.2.3, the input ECG signals were retrieved from real-world recordings, available in the Physionet PAF prediction database [85]. However, in comparison to the previous research, this setup includes 300 recordings, each of 30 min, thereby covering a wider input set. Every recording consists of data acquired from two simultaneously operating ECG sensors, and the root mean squared value of the data from the two sensors. I have considered time windows of 6 min, with an overlap of 5 min, for processing the input data, thereby obtaining 25 time windows as a result for each of the 30 min-long recordings. 6-min windows are preferred in this work as compared to Section 2.2.2.3, as it processes more ECG per window, thereby being more capable in detecting issues compared to smaller time windows. Also, an overlap of 5 minutes helps to maximize the amount of information transferred between windows when consecutive ones are processed.

Every window in each recording was independently processed by the PSA application to evaluate their LFHF ratio. In order to calculate the total energy consumption of the *Extr_buffer* and *DWT_buffer*, the formula in Equation 2.3 was reused, where the number of accesses were derived from a software counter at the application level, and the energy per access and leakage power were extracted using CACTI. To calculate the IM dynamic read energy, I have considered a worst-case scenario in which an instruction is fetched at every clock cycle. Binary error masks, similar to the method described in Section 2.2.2.3, were employed in order to determine the erroneous bits in the unprotected part of the memory. Upon occurrence of any error in the unprotected part of the *DWT_buffer*, the whole word was set to the statistically expected value of zero. The application runtime was evaluated by running the PSA algorithm

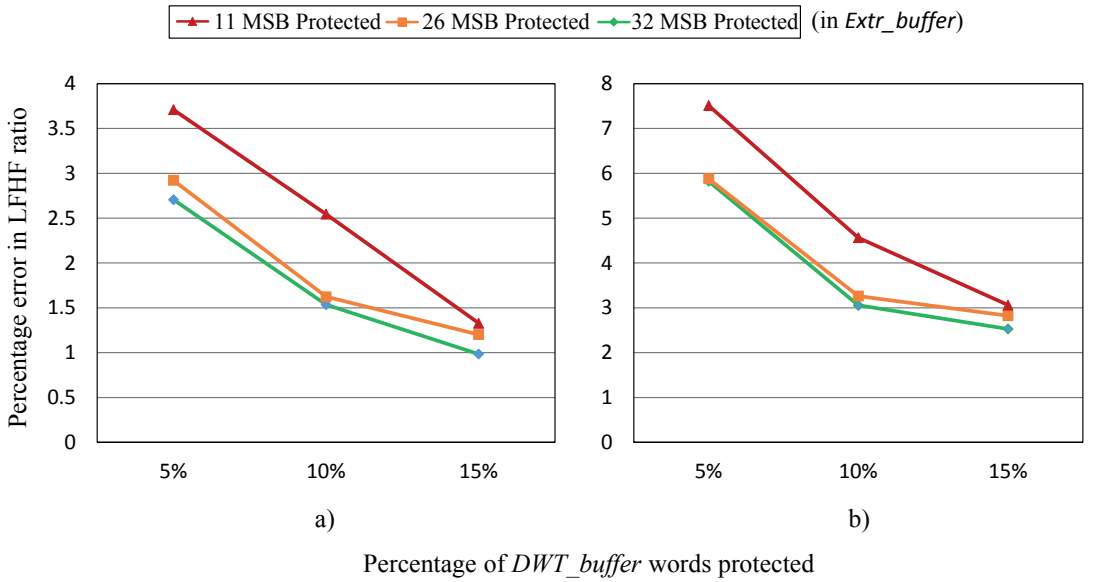


Figure 2.11 – Percentage error in the LFHF ratio under the different memory protection schemes: a) at 650 mV and b) at 600 mV.

on the target ARM Cortex M3 processor [90], and was found to be 2.23 s. Finally, similar to the previous study, realistic ultra-low supplies of 650 mV and 600 mV have been considered, with associated single bit-flip probabilities of 0.07 % and 0.22 %, respectively [35].

The impact of these errors on the quality of the output of the PSA application, under the different protection schemes, was then measured by comparing the obtained LFHF ratio to an error-free execution. To evaluate the effectiveness of the proposed architecture, it was compared against two different baselines:

- ***High V_{dd}:*** In the first case, I considered a high supply voltage (1.1 V), which does not impact the reliability of the system memory. All memories in this case were implemented as 6T SRAMs, whose energy values were computed by modeling them with CACTI.
- ***Low V_{dd} and total ECC protection:*** In the second case, I have considered exact operations at a low supply voltage level (650 mV). This condition requires the implementation of the IM with 8T SRAMs, while all of the DM (buffer and non-buffer) is completely protected by SECDED ECC codes.

Error analysis

The results of the error simulations have been achieved by aggregating the individual results obtained by processing of data in each time window for each ECG recording. Figures 2.11.a and 2.11.b show the percentage of error in the computation of the LFHF ratio by the PSA application when compared to an error-free version of the same, under the different test-points

of the proposed heterogeneous memory scheme, at supply voltages of 650 mV and 600 mV, respectively. Obtained results showcase that selective protection of a small fraction of words (the significant ones) in the sparse buffers can still guarantee high-quality performance of the system with respect to an error-free version, thereby showing the error-tolerance capabilities of BSP applications. As an example, only 1.3 % relative error is incurred in the LFHF ratio by protecting 11 MSBs in the *Extr_buffer* and 15 % significant words in the *DWT_buffer* (Figure 2.11.a). In clinical practice, this error magnitude can be well tolerated, as medical literature reports results on LFHF analysis with only 2–3 significant digits [91].

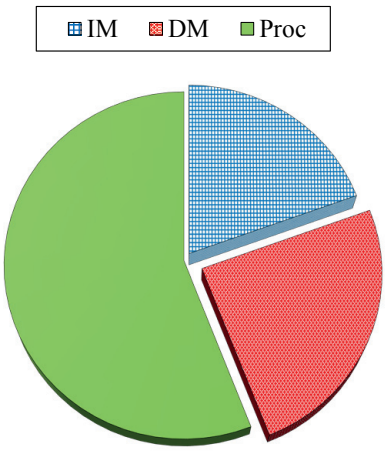
The percentage of error in the LFHF ratio obtained by protecting only 4 MSBs in the *Extr_buffer* exceeded 20 % even in the case of full protection of the *DWT_buffer*, showing the importance of MSB protection for non-sparse data. This high error-rate is however not acceptable in the BSP domain and the condition of protecting only 4 MSBs in the *Extr_buffer* is thus excluded from my experiments.

When 32 MSBs are protected in the *Extr_buffer* and 15 % of significant words are protected in the *DWT_buffer*, the relative error is less than 1 %. This figure is bounded below 4 % even when the worst-case protection from the experimental setup (11 MSBs protected in the *Extr_buffer* and 5 % of significant words protected in the *DWT_buffer*) is considered. The percentage error in the LFHF ratio for a supply voltage of 600 mV is shown in Figure 2.11.b, with the same trends observed as in Figure 2.11.a but with higher relative error with respect to operation at 650 mV supply voltage. This fact is due to the much higher number of bit-flip errors in the memories at 600 mV supply, when compared to 650 mV. Interestingly, even in this case, the error in the LFHF ratio can be restricted to less than 5 % when compared to a fault-free execution, by allowing errors in the 21 LSBs of *Extr_buffer* and only checking (but not correcting) errors in 90 % of *DWT_buffer*.

Energy analysis

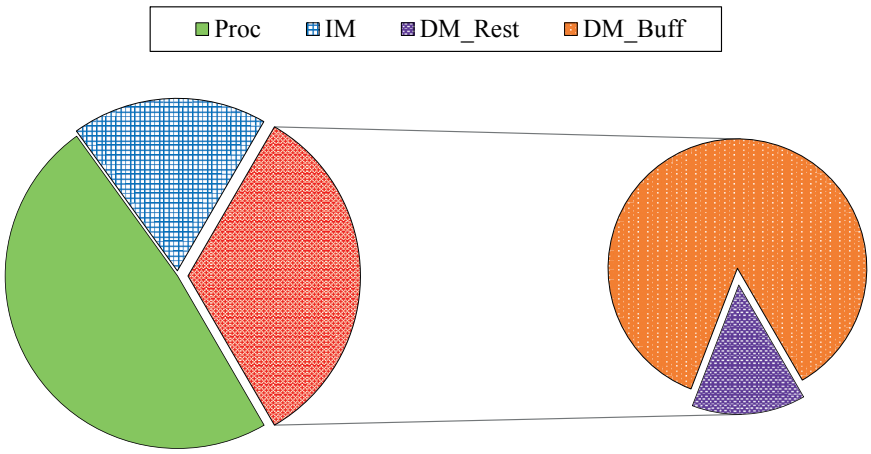
Figures 2.12 and 2.13 compare the energy consumption of the different system components at the first and second baselines considered (high V_{dd} = 1.1 V and low V_{dd} = 650 mV with complete memory protection, respectively). At low supply voltage, the system shows substantial energy savings when compared to operation at high supply voltage. Moreover, it can be noted that the DM accounts for a significant part of the energy budget and that the targeted buffers constitute the majority of the energy overhead of the DM (Figure 2.13). This observation justifies the application of the proposed memory protection scheme to save even more energy in these buffers, thereby further enabling energy benefits at low-voltage operating points.

Figure 2.14 shows the total energy consumption of the targeted memory buffers under the different protection schemes. It can be observed that by using the proposed scheme where 11 MSBs are protected in the *Extr_buffer* and 10 % significant words in the *DWT_buffer* (corresponding to a relatively low error of 2.6 % in the LFHF ratio as in Figure 2.11.a), it enables as much as 18 % savings in the energy in the buffers compared to the second baseline. These



Energy consumed by different parts of the system at 1.1V
Total energy consumed = 581619 µJ

Figure 2.12 – Total system energy consumption at baseline 1 (high V_{dd}).



Energy consumed by different parts of the system at 0.65V
Total energy consumed = 261805 µJ

Figure 2.13 – Total system energy consumption at baseline 2 (low V_{dd} with complete ECC protection).

gains, in turn, bring about an overall 5.2 % increase in energy efficiency for the whole system. It can be further observed from Figure 2.14 that by using the proposed heterogeneous memory protection scheme it is possible to achieve almost up to 20 % reduction in the energy budget of the targeted buffers in the most energy-efficient case of protection considered (11 MSBs in *Extr_buffer* and 5 % of significant words in *DWT_buffer*), over a scheme which involves protecting the buffers completely with SECDED codes. This study therefore points out that by employing memory protection schemes judiciously, it is possible to achieve substantial energy savings at both the memory and system levels.

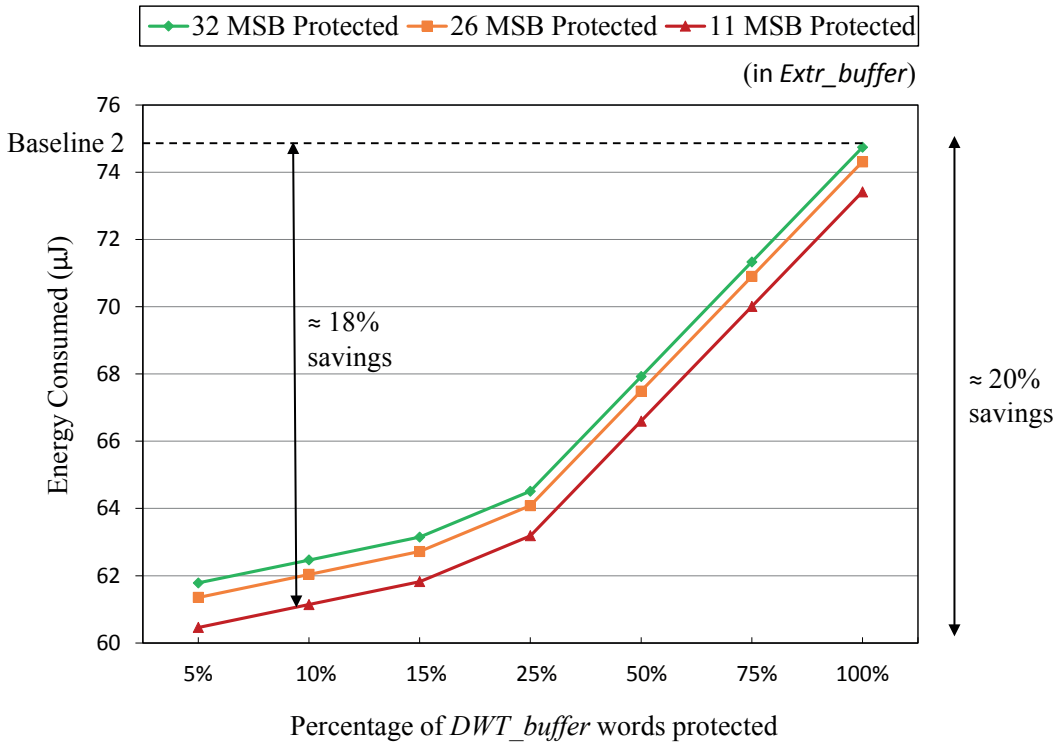


Figure 2.14 – Total energy consumption by the targeted memory buffers at 650 mV supply under different memory protection schemes for an execution time of ≈ 2.23 s.

Energy/quality-of-service trade-offs

Combining the results obtained in the previous sections, it is possible to investigate the trade-offs between relative error in LFHF ratio and the energy consumption. This study enables the selection of the optimal memory architecture considering both energy consumption and performance degradation. In Figure 2.15, the total energy consumed by the targeted buffers under the different protection schemes are plotted against the corresponding percentage error in the computation of the LFHF ratio at 650 mV. It highlights that protection schemes with less energy consumption result in higher performance degradation. As an example for selecting the optimal protection scheme, for a maximum tolerable error of 3 %, a solution with 11 MSBs protected in the *Extr_buffer* and just 10 % of the most significant words in the *DWT_buffer* is the best one in terms of energy efficiency. On the other hand, for an assumed energy budget of 62 µJ, the best output quality can be achieved by protecting just 32 MSBs in the *Extr_buffer* and 15 % of the *DWT_buffer*.

This research therefore shows the energy saving possibility by judiciously protecting the memory subsystem, enabling the selection of the most suited type of protection. The proposed approaches can also be extended to other application domains where energy efficiency is of paramount importance while inexactness can be tolerated to a certain extent.

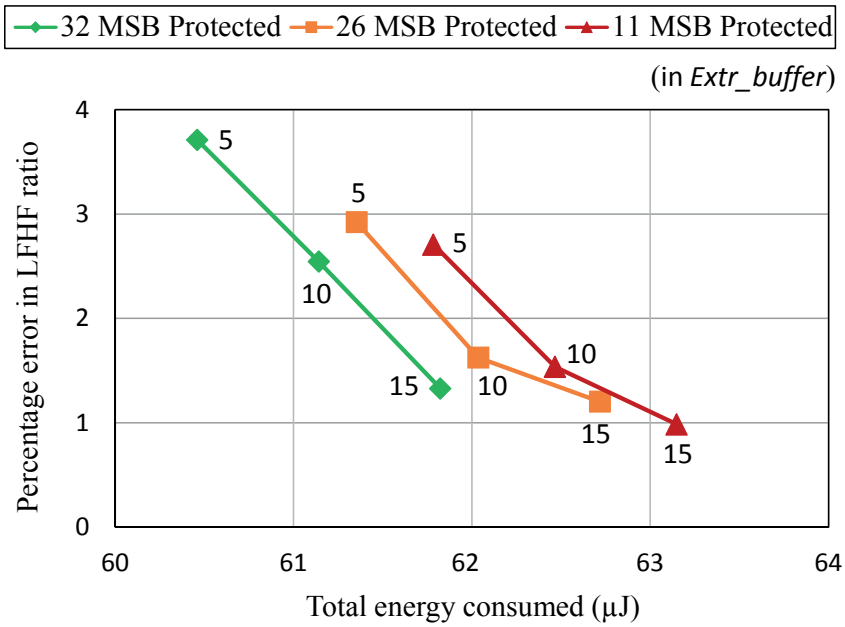


Figure 2.15 – Percentage error in LFHF ratio and the corresponding energy consumption under different memory protection schemes at 650 mV. Numbers beside the points represent the percentage of *DWT_buffer* protected.

2.3 Relaxed-reliability computation with error management

2.3.1 Introduction

Voltage over-scaling can reap significant benefits in energy-constrained low-power WBSNs by drastically cutting down their power consumption. However, ultra-low voltage scaling can lead to errors in memories due to dynamic and static variabilities. Regarding the former, the availability of constant supply voltages cannot be warranted due to voltage fluctuations (*droops*) occurring at runtime. The study of effects of voltage supply droops is thus very important, especially at ultra-low values, since they can lower the effective supply voltage to Near-Threshold Voltage (NTV) ranges of transistors from time to time, at which SRAMs tend to become unreliable. Moreover, fabrication-time process variations also impact SRAM reliability since some cells may be more prone to errors than others. Therefore, herein I also consider the effects of runtime voltage supply droops and fabrication variabilities in SRAM design and operation.

Due to the unreliability of SRAMs at NTV, operation at such voltage ranges results in the imposing of undesirable high energy-consuming guard-bands to guarantee correctness under worst-case conditions. In this research I take an alternative stance: by applying relaxed-reliability computation, I allow failures to occur, abandoning the necessity of additional memory protection and guard-bands for insuring steady voltage supply. Instead, I adopt lightweight mechanisms at the hardware and software levels of the Bio-Signal Processing (BSP)

system that can ensure proper execution at scaled down voltages, thereby entailing important energy savings.

The rationale behind this approach is that in WBSN applications, bio-signal acquisitions are always mixed with noise, while BSP outputs are often qualitative or statistical in nature. Consequently, errors can be tolerated if the system is never trapped in an inconsistent state due to a failure, and if it can be ensured that their effects on the produced output will be limited. This methodology bridges the error tolerance exposed by BSP with the energy-saving opportunities deriving from relaxed-reliability computation [92]. It also links the characteristics of nano-scale ICs and those of application scenarios in a cross-layer, hardware/software co-design framework. Relaxed-reliability computation can be enforced through low-overhead hardware/software mechanisms both on-board the WBSN as well as on the receiver side. By trading-off the failure rates with energy efficiency, the goal of this research is to maximize the amount of correct outputs produced by the BSP, for a given energy envelope, while at the same time minimizing the impact of failures.

In this research, conversely to the previous section, I do not employ any dedicated memory protection scheme, permitting the resulting failures (in both Instruction Memory, IM and Data Memory, DM) under low-voltage operations to spread to the processing system. To prevent the detrimental manifestation of such errors on the application output, I propose a set of lightweight hardware/software error monitoring techniques on the WBSN itself that mitigate errors by detecting them and discarding the corresponding corrupted computation.

2.3.2 Effects of voltage scaling on SRAM memories in presence of supply fluctuations

As described in Section 2.1.2, failures of SRAM cells are dependent on the probability of read, write, hold and access failures, which are different for each cell depending on fabrication-time variations. This probability of failure increases as the supply voltage is aggressively reduced [32]. Operating at low supply voltages is aggravated by two main additional challenges: first, the electrical characteristics of Integrated Circuits (ICs) are increasingly difficult to control in nano-scale technologies, resulting in a high level of fabrication-time variability [93]. Secondly, supply voltages, especially at low levels, experience fairly large fluctuations at runtime (called voltage *droops*). These effects become especially challenging when supplies approach the NTV regions of transistors, where SRAM cells become prone to errors. Moreover, the frequency of operation is adversely affected as voltages go down, an effect which impacts both the memory and logic blocks of digital architectures. However, in the BSP domain, as platforms usually adopt a low working frequency, especially when applications' tasks are distributed on multiple cores [46], a low operating voltage can be tolerated.

At these low frequencies, logic components can reliably operate even in the near-threshold region, thereby pushing the reliability bottleneck toward the memory subsystem. Multiple voltage supplies (higher for memories and lower for logic) have been proposed in the literature

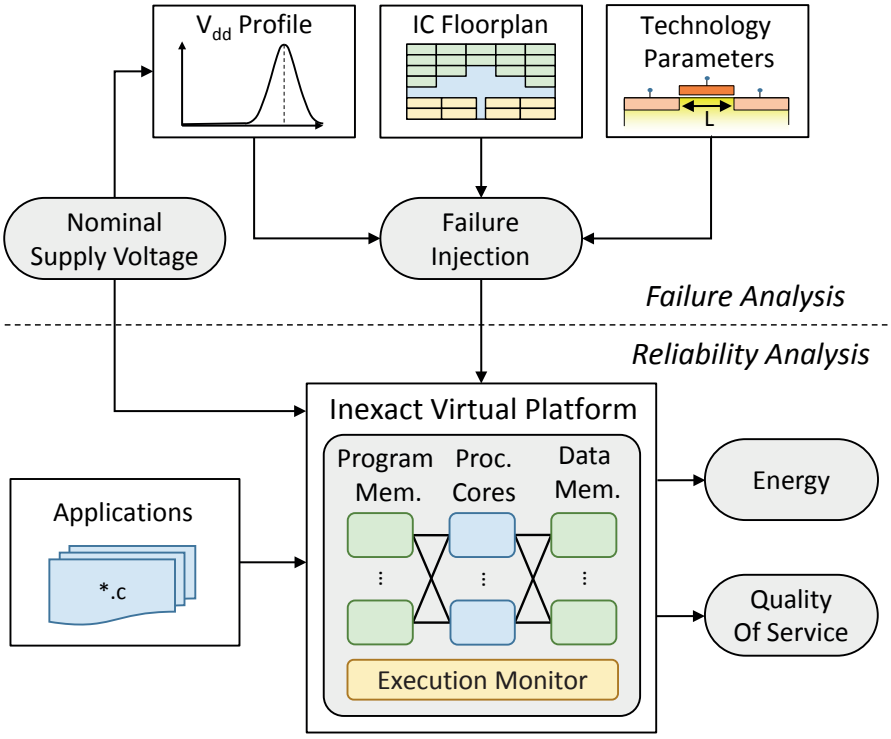


Figure 2.16 – The developed framework analyzes the impact of hardware failures (top) at the application level, when executing on the proposed error-resilient platform (bottom).

to counteract this effect. In fact, the authors of [94] describe an IA-32 processor that exploits the logic resiliency by adopting different supply levels for memory and logic. In [94] it is shown that, below a certain target frequency (100 MHz in their implementation), the memory voltage cannot be further reduced, while further voltage-frequency scaling is applicable to processor components. Such dual-voltage strategy increases the system complexity and area, because multiple supply voltage (V_{dd}) lines must be distributed and voltage converters need to be provided for connections crossing the voltage islands. Single-V_{dd} solutions are therefore better suited to the requirements of resource-constrained embedded processors [24]. However, in certain domains, such as in BSP, since the operating frequency requirements are inherently low, these features make NTV operations an attractive choice for BSP platforms, provided that the memory failures do not translate into any major system failure or output degradation. When instead, voltage supplies fall below the threshold voltage, efficiency gains flatten out, while performance penalties steeply increase.

To assess the impact of variability and voltage droops at runtime, I have adopted a two-step approach, illustrated in Figure 2.16. As the first step, I derive from a nominal supply level (V_{nom}) and a statistical voltage distribution, the failure probability of the target memory structure. In the following step, failure probabilities are linked to application-visible errors, while the nominal voltage is employed to retrieve the energy consumption of the system.

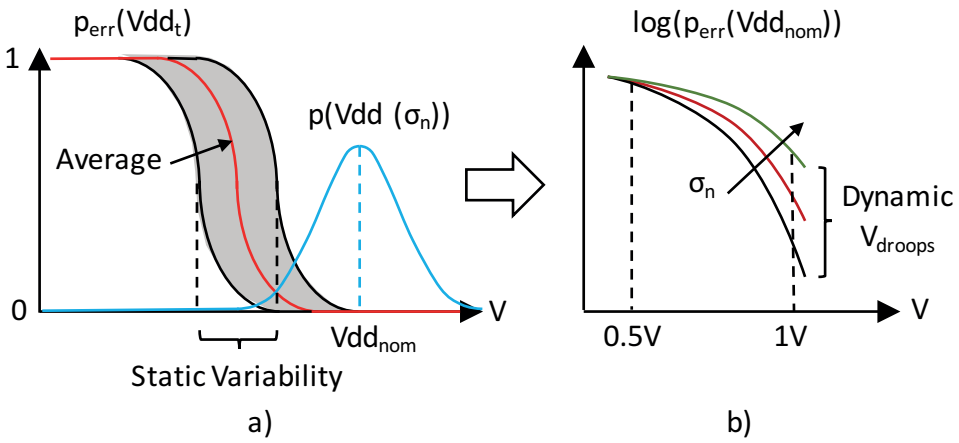


Figure 2.17 – Error rates at nominal voltage (right), dependent on static variability (left, gray area) and voltage noise profile (left, blue line).

2.3.2.1 Memory failure modeling

SRAM failures (bit-flips) at NTV can be caused by various mechanisms [95] (cf. Section 2.1.1). Hold failures happen when the content of a memory cell is lost, due to leakage currents, while not being accessed. Write stability failures occur when the correct value is not stored in a cell, even when the write time is infinite. Read and write timing failures appear when access times are too short to transfer a state to/from a cell. Finally, a read upset erroneously flips a memory content on a read operation. The probability of these events depends on a number of factors: the adopted voltage supply, the implementation of the cells, the memory organization and its static variability. In this study, the target technology is an SRAM memory subsystem with 8-transistor (8T)-based cells. This choice is preferred over a 6T SRAM, since 8T cells have dedicated read-write paths that make them much more resilient to voltage-scaling effects [34], while recent progresses in their development have brought their energy consumption almost at par with 6T cells [35].

As a starting point, failure rates of the different cells of the employed SRAM are obtained considering stability issues and timing violations for an instantaneous supply voltage ($p_{err}(Vdd_t)$). This model takes into account the memory structure and floor-plan of the target architecture (detailed in Section 2.3.3.1): number of memory banks, their sizes, word width, etc. In this way, I examine the effect of static variability on memory reliability, considering multiple (100) variation maps, generated by employing the memory modeling tool VARIUS-NTV [96].

In the following step, the effects of voltage droops are considered by modeling Vdd fluctuations as a normal distribution, centered on Vdd_{nom} ($N(Vdd_{nom}, \sigma_n)$). This formulation is in good accordance with the empirical evidence reported in [97] and [98], while at the same time abstracting the details of power delivery system implementations. A detailed study of the power delivery system, investigating the design of the voltage regulators and the power delivery network is, however, outside the scope of this thesis. The error probability for a given supply

voltage and an associated noise profile can then be derived from the area below both curves, as depicted in Figure 2.17. Therefore, the error rates, when varying the Vdd_{nom}, can be derived as in Equation 2.4:

$$p_{err}(Vdd_{nom}, \sigma_n) = \int_0^{\infty} p_{err}(Vdd_t) \times N(Vdd_{nom}, \sigma_n).dVdd_t \tag{2.4}$$

In a traditional error-free exact architecture, all memory faults must be avoided, requiring a voltage supply level such that $p_{err}(Vdd_t) = 0$, under all variability conditions. By adopting relaxed-reliability computation, however, a non-zero fault probability is instead tolerable at the system level. It allows to focus on the average dependency between faults and the voltage supply, instead of a worst-case one.

2.3.2.2 Application-level error manifestation

Bit-flips occurring in the target SRAM memories can affect both the execution of an application, as well as the outputs that are produced by it, leading to their degradation. As opposed to the study in the previous section where the IM and part of the DM were protected against failures, herein I have considered errors in the entire memory subsystem, without employing any protection scheme. Thus errors in the IM are also taken into account that can lead to inconsistent system states. The result produced by a simulation with error injection in a memory cell is compared to one without error, and according to the severity of the impact, errors are typically classified into three different categories [99] [71] [100], as described in the following. The detailed classification algorithm that was adopted is depicted in Figure 2.18.

- ***Masked errors:*** A bit-flip does not influence the runtime flow of the application, nor the output or nature of the results. Masked errors are generated when an erroneous bit does not have any significance on the content a word it represents (e.g., unused bits in instruction encodings), or does not have any effect on the result of a computation.

When the output of the simulation with error-injection does not match the output produced without any error in the memories (golden output), one of the following situations can occur: The lengths of the outputs (i.e., the amount of compressed samples produced) matches that of the golden simulation, but one or more values are altered as a result of the injected error. Contrarily, if the lengths do not match, it indicates a probable system crash or hang. The former condition means that although the system did not crash due to the error, some values were changed as a result of the injected error and thereby resulted in a corrupted output. Such errors are classified as Silent Data Corruption (SDC) errors, described in the following.

- ***Silent Data Corruption (SDC) errors:*** In the presence of a bit-flip, from a runtime point of view, the application does not suffer any major change and it produces the correct

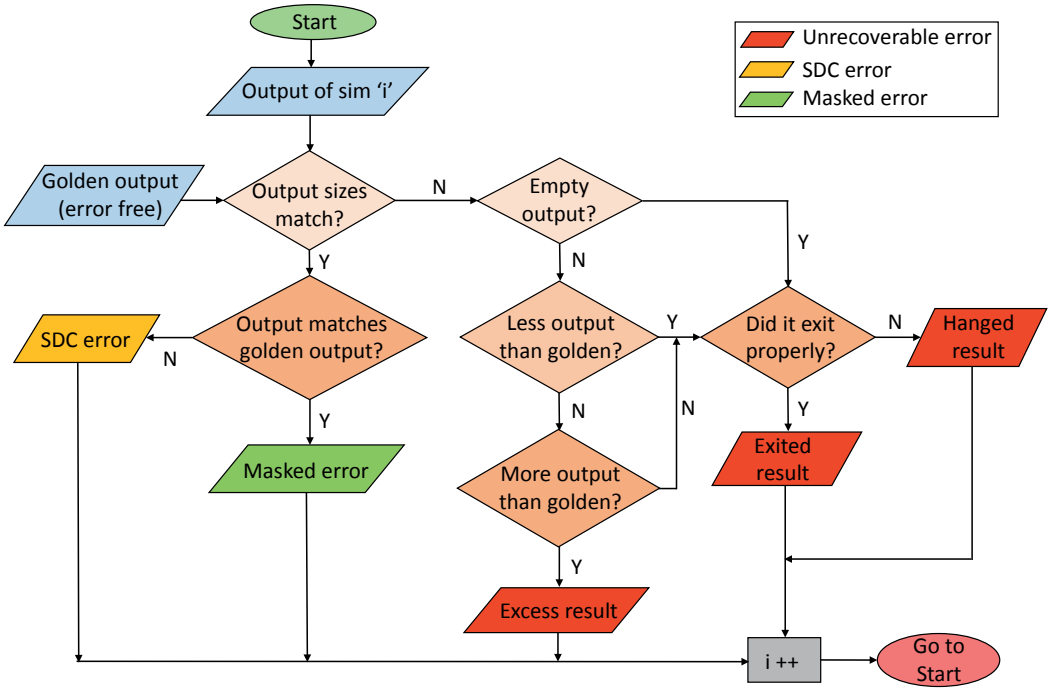


Figure 2.18 – A flowchart showing the application level error classification strategy.

amount of results at the required rate. However, the bit-flip influences the output by corrupting it to some degree with respect to the expected one. While some of these errors can be the result of data modifications (e.g., a bit-flip while a processing core reads a word representing a sample of an acquired signal), others can be produced by changes in the execution flow (i.e., premature exit from a loop iteration, or a function). It is important to note that while some SDCs do not result in system crashes, they may be unrecoverable. These SDCs (like the unwanted modification of values in the read-only section of the DM) must therefore always be avoided.

On the other hand, the output of a simulation with error-injection can produce results whose length do not match that of the expected golden one. In the first scenario, there is no produced output, signifying that the system crashed before the first output value could be generated. In another case, the length of the produced output can be less than the expected one, suggesting a system crash after a few output values were generated. These two cases are results of the situation when the system hangs due to the cores getting stuck as a result of the introduced error, or if the system exits upon encountering an illegal jump to an unspecified IM location (cf. Section 2.3.2.2). Finally, if more than the expected output values are generated, it signifies that the employed cores produced excessive results due to a possible change in a loop counter value, or cores other than the intended ones also erroneously started to execute. All these errors involve system crash, and are therefore classified as unrecoverable errors, as discussed in the following.

- ***Unrecoverable errors:*** A bit-flip leads to a runtime change in the application that either breaks the execution flow, or drastically interferes with its execution. Bit-flips were classified as unrecoverable errors if any of the following situations occurred as a result of them:

  - *Illegal jump:* These are errors that occur when a processing core jumps to an uninitialized region of the IM. To detect such an illegal jump, the unused words of the IM were preinitialized at startup time to a predefined instruction that causes the execution to exit immediately.
  - *Exceeding processing deadline:* If an execution exceeds the real-time processing requirements, it indicates that the cores are taking more time to finish execution than needed (e.g., by getting trapped in an infinite loop or one with a modified number of iterations). These errors result in an inconsistent system state, because as the processor runs unexpected instructions, it finishes processing before/after it should, or its execution is blocked.
  - *Critical memory data modification*: These unrecoverable errors can be the result of bit-flips affecting both the instructions (e.g., a conditional branch changes to be unconditional when evaluating a loop iteration), or the data words (e.g., the content of the link register stored in the stack is modified resulting in a jump to an incorrect memory location).

2.3.3 Experimental setup and results

This section details the experimental setup adopted and the obtained results. First, the target BSP platform employed for evaluation is presented. Then, the proposed error monitoring strategies are described along with their implementations on the target platform. Finally, the obtained results are discussed.

2.3.3.1 Target evaluation bio-signal processing platform

This section describes the target BSP platform that is considered for the application of the proposed error monitoring strategies. First, the low-power processor, which is the building block of the target platform, is presented, and then the employed architecture is elaborated.

A low-power bio-signal processor: TamaRISC

As shown in Figure 2.19 [101], the TamaRISC is a simplified Reduced Instruction Set Computer (RISC) Domain Specific Instruction-set Processor (DSIP) architecture aimed at processing BSP applications [101], and is employed in this research work. It provides a highly reduced set of instructions that cover all the typical computational requirements of BSP, thereby reducing the hardware complexity of the platform. It features a Harvard-like architecture with a three-stage

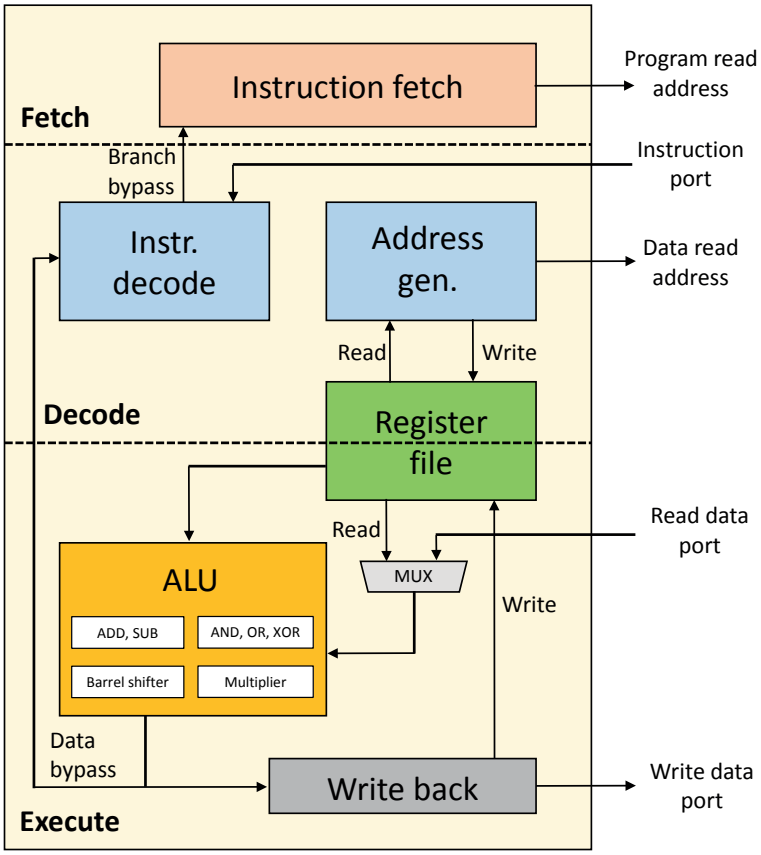


Figure 2.19 – Block diagram of the TamaRISC architecture.

pipeline consisting of fetch, decode and execution stages. It has a 16-bit wide datapath and a local register file of 16 registers (each of 16 bits). The processing core is interfaced with the IM though a read port and with the DM through two ports, one dedicated for reading and the other for writing. The TamaRISC instruction set includes 8 Arithmetic Logic Unit (ALU) operations, 2 program flow instructions and 1 data move operation. The ALU instructions consist of addition and subtraction, with and without carry, logic operations (AND, OR, XOR), left and right shifts (arithmetic and logical) and 16-bit unsigned multiplication (with result in 32 bits). Both direct and register indirect modes are supported in the general branch instruction. It also supports offset with respect to the program counter, and can be executed conditionally depending on the carry, zero and overflow flags. Moreover, a “CALL" instruction is also supported that is used to perform function calls, and is implemented as a “branch and link" to a fixed address.

In the TamaRISC architecture, all instructions take a single clock cycle to execute, due to the bypassing of data from the execution to the decode stage that allows single-cycle memory-to-memory instructions. The hardware complexity of the architecture is also reduced significantly thanks to the 24-bit instruction encoding that enables a simplified decoding hardware. Moreover, the ALU operations in the execution stage receive three operands that are obtained

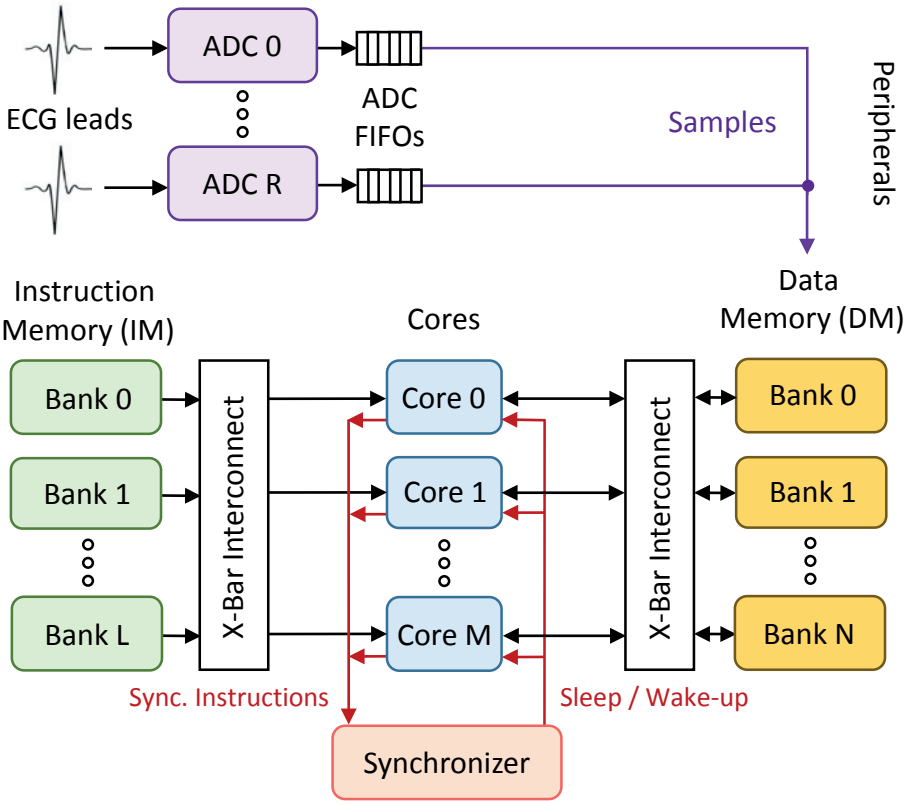


Figure 2.20 – The target bio-signal processing platform.

by using the same addressing modes. This method further reduces the complexity of the logic associated to this task. As a result the TamaRISC architecture consumes on an average 17.1 pJ of energy per instruction (at 1.2 V supply, 65 nm technology) [101], which is significantly smaller than state-of-the-art DSIPs. In addition, it supports voltage scaling, thereby potentially increasing the energy efficiency even more.

Multi-TamaRISC target platform

In order to minimize the effect of bit-flip errors in the SRAM memory subsystem, I propose a few lightweight strategies which lie on both hardware and software sides. I start from a SoA multi-core BSP platform as the target, featuring eight TamaRISC processing cores as shown in Figure 2.20. The system is fed with ECG signals extracted from multiple leads, which are first digitized and stored in the eight ADC First-In-First-Out (FIFO) buffers. These data are then fed to the core that is responsible for processing the corresponding signal. The instructions needed to run an application are imported into the Instruction Memory (IM) during system startup, typically from a flash-based storage, which is then switched off. The IM has 8 banks, each of which is private to a processing core. The Data Memory (DM) stores the data fetched from the ADC FIFOs and also the intermediate data that is produced during processing. The

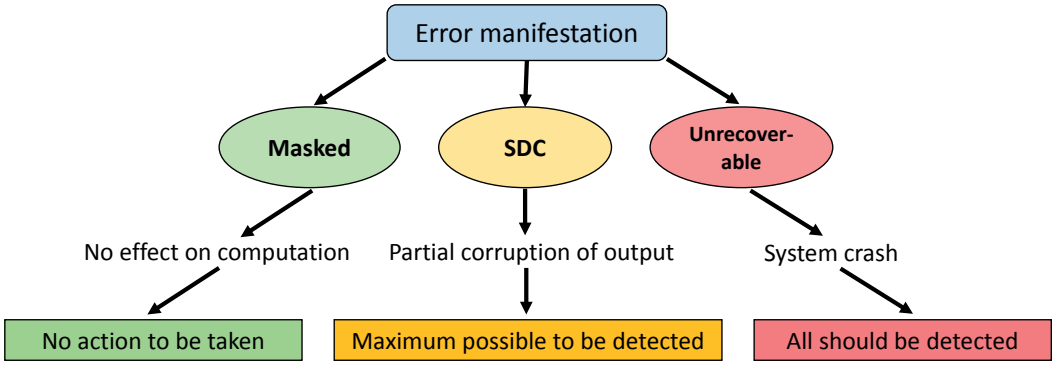


Figure 2.21 – The proposed error classification strategy along with the corresponding action to be taken in each case.

DM has 16 banks, partitioned in a shared section devoted to inter-processor communication, and private sections for each core. It also contains a read-only part that embeds application-specific data (constants, parameters, etc.) that should not be altered during the course of the application-processing. The memories are interfaced with the processing cores through dedicated crossbar interconnects (X-bars), which are able to coalesce memory requests when the same data or instruction is read by multiple cores in SIMD mode [46]. Finally, the platform embeds a *synchronizer* module, that monitors the execution of the multi-core system, and does code synchronization. It is responsible for restoring cores from runtime divergences, like branches, and resumes SIMD execution. In addition, it also coordinates the IM accesses (avoiding conflicts) and clock-gates the cores that are not performing any active computation.

2.3.3.2 Error monitoring and recovery strategies

This section details the error monitoring strategies and their implementations put in place in the target system, in order to recover from memory errors. In order to mitigate the effects of bit-flip errors in the memory subsystem on the application execution and output, I propose lightweight error-monitoring schemes. The goal of such a strategy (depicted in Figure 2.21) is to detect all unrecoverable errors on the WBSN node itself, and to appropriately handle them to ensure the resumption of system execution from a coherent state. It also aims to detect and cope on-board with a high number of SDC errors, by monitoring anomalies in the execution flow arising due to an SDC error. In my work, I have chosen a pessimistic approach of effectuating a system reset, thereby discarding the processed data, upon detection of an error. However, such an approach potentially ensures that computation will resume correctly afterwards and that the effects of the errors will be limited in time. It should also ensure no system execution changes occur when there are no errors, or when a masked error happens. For SDC errors that might not be detected by the on-board strategies, depending upon the application, some degree of filtering on the receiver side may also be possible.

Error symptoms and monitoring strategies

The approach to error mitigation monitors critical system components and runtime events while processing, as well as the computation outcomes, discussed in the following:

- ***Runtime coherence:*** BSP systems, especially when featuring multiple cores, must implement a policy to synchronize their runtime execution. Synchronization events, in turn, must follow a legal pattern (e.g., a processor should be notified to resume its execution due to an event only if the core was waiting for it). An illegal synchronization sequence can only be the result of an unrecoverable or an SDC error, which resulted in an incorrect execution behavior.

- ***FIFO surveillance:*** Data is exchanged between peripherals (e.g., ADCs) and processors, or between processors, through FIFOs, which are sized to meet the real-time constraints of the intended workloads. Nonetheless, due to an error, it is possible that a FIFO overflows or underflows. Overflows could happen if a processor is not able to read from the FIFO (e.g., it got stuck in an infinite loop) or an incorrect and higher amount of data is being written into it. On the other hand, underflows can occur if a producer core (possibly stuck in a loop) writes an inter-processor FIFO much slower than the rate at which the consumer core reads it. Overflows and underflows can therefore be used as symptoms indicating the occurrence of an unrecoverable or an SDC error.

- ***Real-time constraints:*** WBSNs are expected to process data and output the results in real time. If the computation is not finished within an acceptable time window, it is indicative of a possible unrecoverable or SDC error.

- ***Memory protection:*** Some portions of the memory, typically the application read-only section of the DM, should never be written to. These memory sections typically store constant parameters that must not be modified during execution. A write request to these memory sections signals a forbidden memory access, generated by an error in the addressing mechanism (either in the instruction or in the processor stack that resides in the DM).

- ***Detection of IM access violations:*** Failures can cause an incorrect jump to an uninitialized memory location. Accesses to addresses outside of the IM portion of an application can be monitored to detect the occurrence of an unrecoverable error.

- ***Receiver-side detection of erroneous results:*** After wireless transmission, further error detection strategies can be performed by the receiver. For example, an effective approach, adopted in this study, consists of comparing the frequency spectrum of the received data against the expected one, having an a-priori knowledge of the application-admissible values. Such data integrity analysis must be performed without any intervening error, and therefore cannot be executed on the inexact WBSN.

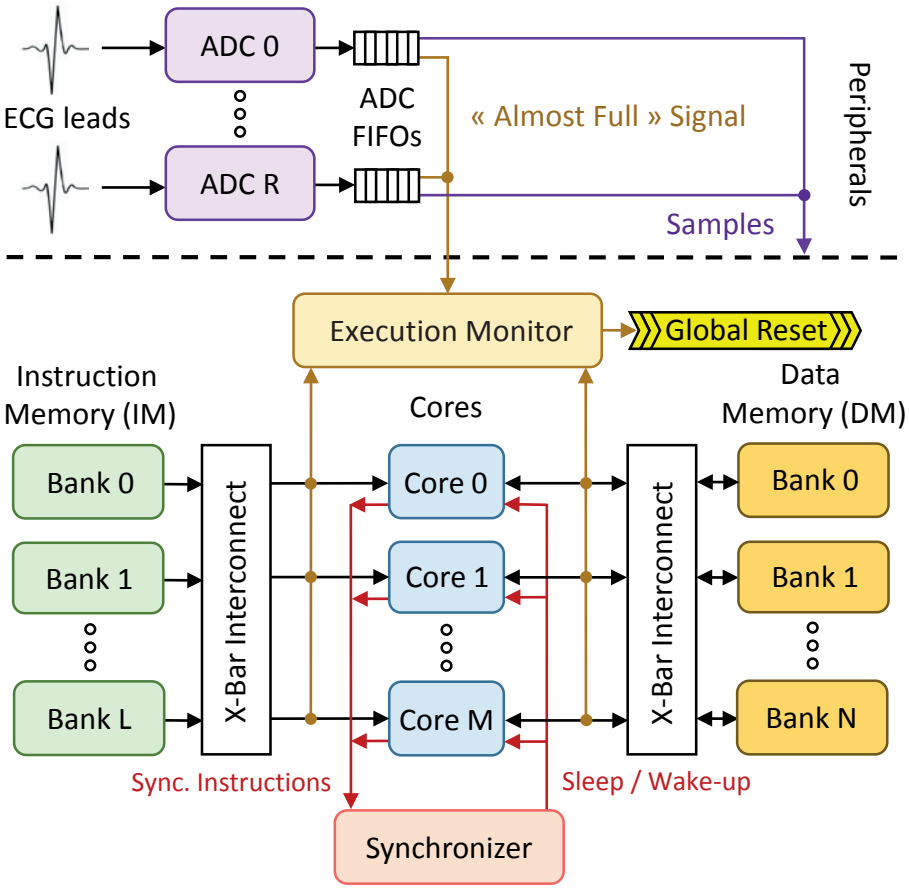


Figure 2.22 – The target bio-signal processing platform with the proposed error-monitoring strategies.

Implementation of the error-monitoring strategies

The error-monitoring strategies described in the preceding section can be implemented incurring a small hardware overhead, by monitoring synchronization events, FIFO states, and IM and DM accesses. I have implemented these strategies in a hardware *Execution Monitor (EM)* module, embedding it in the considered target architecture shown in Figure 2.22 [101]. It monitors synchronization instructions issued by the processing cores to look for possible breaking of the expected trends due to an error. It also keeps track of the FIFO accesses and gets notified of their states though employed *Almost Full* and *Almost Empty* signals for each FIFO buffer. Furthermore, it also checks the IM and DM accesses from the processing cores. In particular, the implementation of the error-monitoring strategies are described in the following.

- ***Runtime coherence:*** Synchronization instructions, managed by a hardware synchronizer component, are employed in the multi-core platform in [12] to manage SIMD execution and producer/consumer relationships between cores. In an error-free exe-

cution, synchronization events are processed in a last-in-first-out order, as a processor can only resume its execution from the last-entered synchronization point. In order to check coherence at runtime, a small hardware stack (of 32 words in my implementation) is provided in the EM for each processor, storing the synchronization point IDs. A pop request with a different ID with respect to the one at the top of the stack is therefore an indication of an error, and causes a system reset.

- ***FIFO surveillance:*** ADCs typically embed FIFOs, where samples are queued when they are acquired, until a processor consumes them. Software FIFOs are also employed to store the intermediate results generated by the producer cores, before the consumer cores use them. In this approach, both memory structures are monitored at runtime by two slightly different mechanisms:

  - *ADC FIFO monitoring:* A signal is raised from each of the 8 ADCs in the system to the EM module to inform it that the corresponding ADC FIFO is almost full. For this *Almost Full* threshold, I have chosen it to be 3/4th of the total ADC FIFO size, which is potentially sufficient when processors execute as expected, since their processing speeds are much higher than the ECG acquisition rate. Upon triggering of this signal, the EM, in turn, issues a system reset.
  - *Internal FIFO monitoring:* A similar strategy employing *Almost Full* and *Almost Empty* signals is also used for monitoring the internal FIFOs situated between the producer and the consumer cores. The EM stores in a register the value corresponding to the number of words contained in each FIFO. A write request to an almost full FIFO or a read request to an almost empty one results in a system reset. The chosen threshold for *Almost Full* is $3/4^{\text{th}}$ of the FIFO size, while that of the *Almost Empty* signal is $1/4^{\text{th}}$ of the buffer size, which has been found suitable comparing the acquisition and processing speeds.

- ***Watchdog timer:*** This detection method, implemented globally on the system, ensures that the applications meet their real-time processing constraints. A reset signal is issued by the EM if the computation is not finished even after an allowed period of time is exceeded starting from the expected end time of the processing (I assumed a 3 s limit for a computation time of ≈ 2 s, cf. Section 2.3.3.3).

- ***Lightweight memory protection:*** The goal of this error detection mechanism is to ensure that the read-only section part of the DM is never written into. For each core, 128 words are allocated for storing the read-only data, starting from the beginning of the private memory section. Programmers can specify a variable to be read-only during the linking phase of the source code compilation. At runtime, the EM module checks for write requests to the read-only section of the DM. Such a mechanism ensures that constant application parameters are never modified, which is crucial to ensure that errors are never propagated after a reset event.

- ***Detection of IM footprint violations:*** Some unrecoverable errors are related to changes in the execution flow that make the cores jump to an incorrect IM address corresponding to an un-initialized memory location. To tackle this issue, all the IM words are pre-initialized to an instruction which forces the processor to branch unconditionally to a predefined error-handler instruction. This instruction is then used to notify the EM to immediately trigger a system reset.

In addition to error monitoring checks on the WBSN itself, it might be possible in certain scenarios to perform a check on the receiver side. For example, in ECG monitoring, the shape of the received and possibly corrupted ECG waveform can be compared to the expected one, in order to determine the degree of corruption. This situation is aided by the fact that ECG signals are sparse and inherently noisy, thereby showing tolerance for some additional noise. In the cases where this value is more than an acceptable clinical threshold when compared to an error-free case, the corresponding received signal must be rejected, while those with a corruption value less than the threshold might still be admitted. Considering the addressed application of signal filtering and compression, the employed receiver-side monitoring strategy is detailed as follows:

- ***Receiver-side detection of erroneous results:*** The impact of SDC errors can be further minimized at the receiver side by analyzing the power-spectrum of the received data, and filtering out windows of signals that do not correspond to the characteristics of an ECG acquisition. In order to do so, I have employed a two step filtering process: I first compare the power residing in the frequency range of 0-20 Hz (named P_{f20}) and the power in the full signal (P_{full}), discarding signals with (P_{f20}/P_{full}) < 0.9. Then, a second level of verification is carried out, where the power in the frequency range of 0-10 Hz (P_{f10}) is compared with P_{f20}, discarding the signals if the ratio (P_{f10}/P_{f20}) < 0.1. This technique was validated by processing 1080 random windows of 1024 samples from the MIT-BIH Normal Sinus database [102], achieving a rate of false positives (correct windows classified as corrupted by SDC errors) of less than 1 %.

2.3.3.3 Experimental results

In this section, I showcase the results that have been obtained as an outcome of the application of the proposed relaxed-reliability computation scheme with lightweight error recovery. This section is structured as follows: first, I introduce the benchmark BSP applications that I have employed to evaluate the efficiency of the proposed scheme, followed by the experimental setup. Then, I analyze the application-level errors which occur due to memory failures under voltage over-scaling, in presence of runtime droops. Next, I evaluate the effectiveness of the proposed error-recovery strategies and the associated overheads incurred. Finally, I conclude this section by studying the trade-offs between the Quality-of-Service (QoS) degradation and the energy savings, under the different supply voltages considered.

Target bio-signal processing applications

To evaluate the effectiveness of the error monitoring strategies (described in the preceding section), in countering memory failures arising out of effects both from variability issues and runtime voltage droops, and to investigate the energy efficiency/QoS trade-offs, I have used two SoA BSP applications: one which is embarrassingly parallel, and another which shows a producer-consumer relationship. These two applications, as described below, cover both of the following aspects: (i) when cores can operate independently, as well as (ii) when they are inter-dependent while processing.

- ***Compressed sensing (CS):*** An efficient strategy to reduce the amount of data transmitted from a WBSN through the energy-hungry radio-link is to perform on-node BSP, to filter-out the significant data to be transmitted. Since bio-signals like ECG consist of sparse data, in this scenario, an effective solution to reduce data transmission is to compress the acquired data on-board. Such an algorithm, known as Compressed Sensing (CS), which takes advantage of the sparseness of bio-signals, has been proposed in the field of ECG processing [13]. It compresses the input ECG by using a sensing matrix, as described in Equation 2.5:

$$y = \phi x \tag{2.5}$$

  where $x \in \mathbb{R}^n$ is the input vector of ECG samples, $\phi \in \mathbb{R}^{k \times n}$ with $k < n$ is the *sensing matrix* and $y \in R_k$ is the resulting compressed vector. By performing the vector-matrix multiplication, ϕ maps the input vector x into y with a compression ratio of n/k. By using a high compression ratio, the amount of data to be transmitted can be drastically reduced. Although the reconstructed (decompressed) signal may suffer quality loss [103], it has been shown that ECG signals compressed by 50 % can be reconstructed with a very good signal quality, therefore presenting a good trade-off [13]. The main computational complexity of such an algorithm lies in the matrix multiplication function, shown in Equation 2.5. In particular, the main overhead of this algorithm is the high memory footprint due to the typically large size of the sensing matrix. For example, assuming a 50 % compression ratio for a window of 1024 ECG samples, the necessary random sensing matrix would require up to 2 megabytes of memory for storage. Nevertheless, in [13] it has been shown that choosing a proper random sparse sensing matrix with few non-zero components per column leads to a low-complexity implementation, which still preserves the compressed data integrity, making a good signal reconstruction possible. Moreover, by forcing the non-zero components to a value of '1' and by fixing the number of ones per column, the memory space occupied by the sensing matrix can be significantly reduced. The CS application is a fully parallel one with no dependencies between multiple ECG inputs and doesn't produce any intermediate results that need further processing. Therefore, for this application, the processing of ECG from

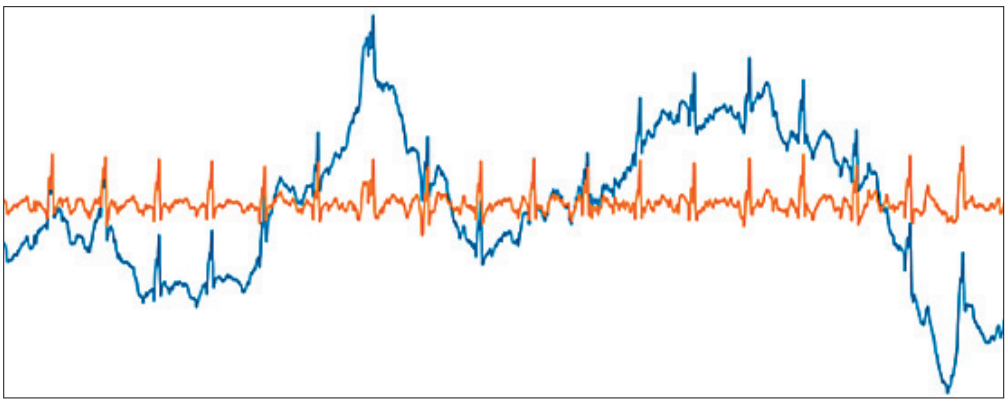


Figure 2.23 – Baseline wander in ECG. In blue: acquired raw ECG with baseline wander, in orange: ECG with baseline wander removed.

each input lead can be done by a separate processor, when a multi-core architecture is employed.

- ***Morphological filtering combined with CS (MF-CS):***

  The other ECG processing application that I have considered is a combination of Morphological Filtering (MF) and CS [100]. Herein, I first describe MF, and then explain the resulting MF-CS application.

  - *Morphological Filtering (MF):* As described in Section 1.1.2, ECG signals are corrupted by baseline wander and various types of noise coming from other bodily activities/organs. Therefore, the conditioning of ECG signals to remove external noise and to extract *clean* ECG is a fundamental application for WBSNs. Such a clean ECG signal can then be used for further compression or processing. In this context, Morphological Filtering (MF) is a well-known technique among various alternatives for suppressing noise in ECG and performing baseline correction [14]. This method involves moderately complex operations on a designated window of ECG signal. It includes the search for the maximum and minimum of the wave window, which in turn helps determine the peaks and pits in the same. The window is then gradually shifted through the entire ECG recording, thereby deriving the baseline wander that is present in the acquired ECG (Figure 2.23). The resulting baseline wander is then subtracted from the input ECG signal in order to produce an ECG that has all waves originating from a common baseline. In addition, the MF application can also be used to filter out the high-frequency muscular noise from ECG by using structuring elements that are shorter than the shortest ECG wave.
  - *Morphological Filtering + Compressed Sensing (MF-CS):* The resulting application first performs the filtering of ECG by employing the morphological filtering algorithm as described above. In the following stage, it compresses the data produced as the output of the MF application. This application shows a producer-consumer

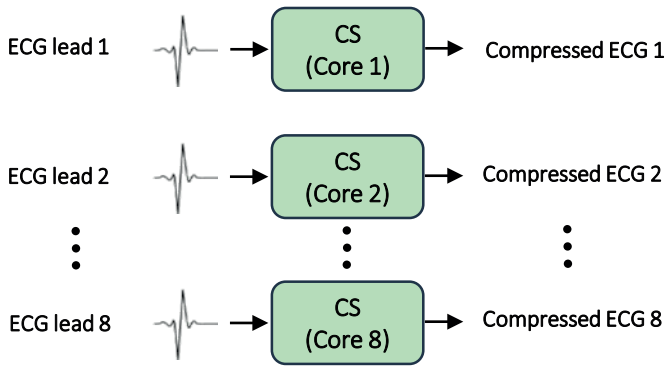


Figure 2.24 – Block diagram showing the mapping of highly parallel CS workloads on multiple processors.

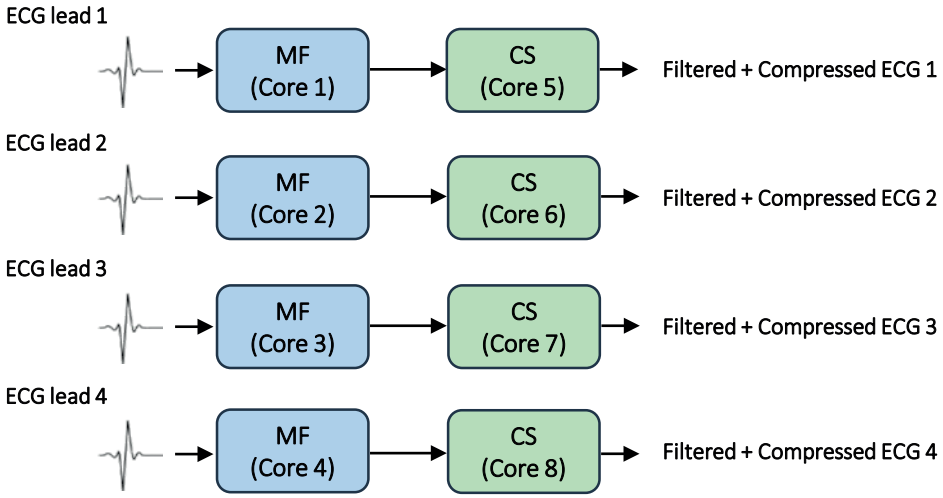


Figure 2.25 – Block diagram showing the mapping of MF-CS workloads on the multiple processors, exhibiting a producer-consumer relationship.

relationship among cores when mapped in a multi-core architecture, where the first set of cores perform the filtering of input ECG leads and the second set of cores compress the filtered ECG, thereby producing the final output of the application.

Framework

For both benchmark BSP applications, I have considered an acquisition frequency of the input signals of 500 samples/sec. The target digital platform comprises eight TamaRISC cores, whose hardware structure and dedicated compiler are generated with Synopsys ASIP designer [104]. The system operates at 5 MHz, a frequency that is sufficient to meet the real-time constraints for the considered benchmarks. Each core, featuring a three-stage pipeline, is interfaced with multi-banked instruction and data memories (implemented as 8T SRAMs) via logarithmic

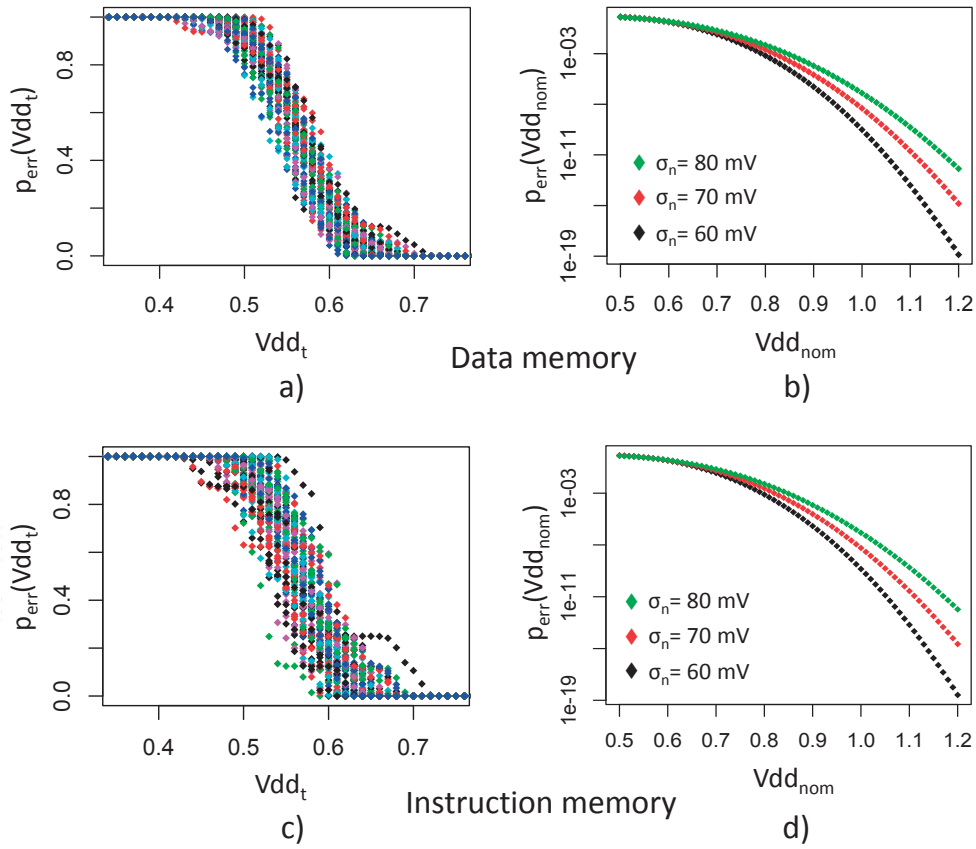


Figure 2.26 – Failure probabilities per bit in the instruction and data memory banks for 100 variation maps considering a fixed supply voltage value (a, c), and in the presence of supply voltage droops (b, d).

crossbars. The DM is arranged in 16 banks of 2 k words each, with a word-size of 16 bits, while the IM is organized in 8 banks of 4 k words, with a word size of 24 bits. In case of a fully parallel application like CS, each core can perform the compression of an ECG signal from a lead, thereby optimizing their usage and maximizing parallel processing. Thus, the execution of CS of each ECG lead is mapped to a different processing core (Figure 2.24). For MF-CS, half of the available cores can be employed to perform the filtering stage while the other half compress the filtered data, thereby displaying a producer-consumer relationship between the two sets of cores (Figure 2.25).

Embedded in the platform, a hardware *synchronizer* module allows execution of code in SIMD mode whenever multiple cores access the same IM location in the same clock cycle [12]. Moreover, it pauses (clock gates) and resumes execution of cores in order to re-establish SIMD execution after divergences due to data-dependent branches and to avoid active waiting when no input data is available (either from the ADC or from another core). Finally, the Execution Monitor (EM) circuitry described in Section 2.3.3.2 overlooks the execution of the different processors, reseting the system whenever an application-visible error is detected.

The system has been synthesized in 65 nm UMC low-leakage technology [105] to retrieve its area and to characterize the energy profile of its components, relying on small synthetic benchmarks. These retrieved parameters were then employed to annotate a cycle-accurate SystemC model of the platform. This strategy enables lengthy simulations of real-world BSP applications, which would be impractically long if performed at the RTL level. The failure rates in the memory subsystem were modeled with VARIUS-NTV [96], according to the methodology outlined in Section 2.3.2.1. SRAM technology parameters (e.g., gate capacitance, channel length of transistors, etc.) were set according to the ITRS road-map [68]. The results, shown in Figure 2.26, consider 100 different variation maps and a Gaussian distribution of the voltage supply with a standard deviation of 60, 70 and 80mV. For each error-injected simulation, a bit-flip has been introduced in the memory system to model failures (either one during an IM read operation or one while a DM address is read/written to). The output generated by the faulty executions of the benchmark applications were then compared to a failure-free execution, retrieving the incurred error type as described in Section 2.3.2.2.

Error analysis

To investigate the effect of memory failures on the application level, multiple simulated executions of the considered applications were performed on the target system. The error recovery mechanism was disabled for these executions as the objective was to let the errors manifest as application-visible failures. Each simulation run processes one ECG window (1024 samples per lead, corresponding to $\approx$ 2 seconds of simulated time), in the presence of a single bit-flip, randomly located in the IM or DM. 3000 tests were performed for the CS application, which covered all the bits in the IM and 3000 non-recurring random bits each in the DM for both read and write operations. Similarly for the MF-CS application, 500 tests were performed as this benchmark requires a considerably larger simulation time. In this way, it was possible to obtain a near-exhaustive study for the CS application, and for the MF-CS it was possible to cover a significant amount of the memory footprint. Each simulation result was then compared to the corresponding error-free application output, thereby enabling their classification into unrecoverable, SDC and masked errors.

The observed distribution of errors types is shown in Figure 2.27. In the CS application (Figure 2.27.a), about a third of the simulations resulted in masked errors, and hence have no application-visible effect. A little more than half of them produced SDC errors, while the rest of the error injection tests create unrecoverable errors, thereby preventing the system from recovering to a correct execution state. Among these unrecoverable errors, a majority of memory bit-flips lead to a jump instruction to an IM location outside of the application footprint, thereby promptly exiting the execution, resulting in a faulty termination. Among the rest, there are either failures that make the processor run into an infinite (or extremely long) loop, ultimately making it miss the real-time execution deadline, or those which cause a wrong application mapping (i.e., when a task is not assigned to the correct core). A similar error breakdown was also observed for the MF-CS application (Figure 2.27.b). For this benchmark,

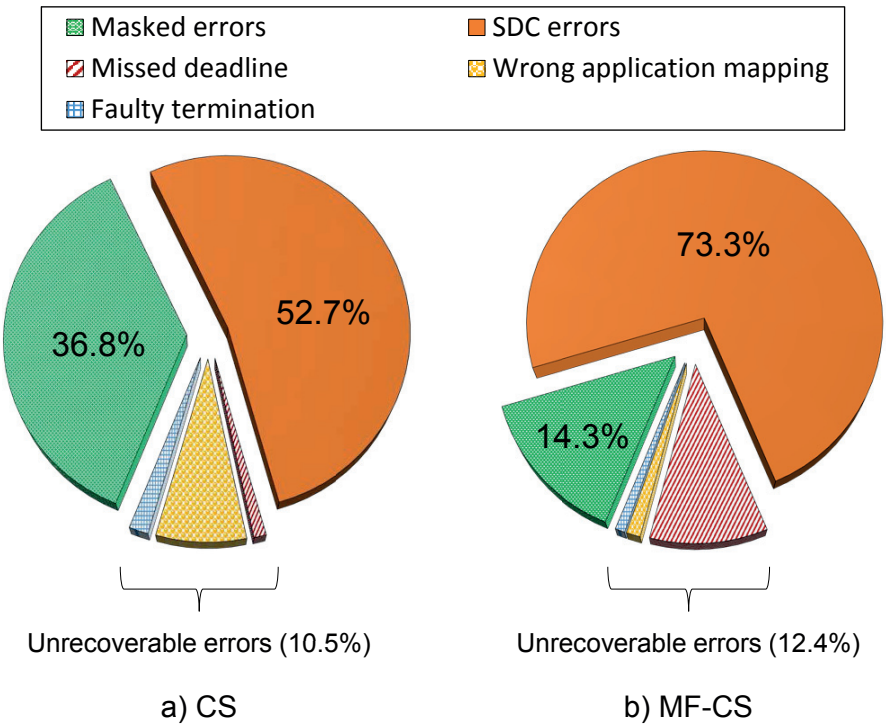


Figure 2.27 – Classification of the resulting application-level errors.

bit-flips causing SDC errors dominate, while masked errors are significantly reduced compared to the case of CS. The number of unrecoverable errors remain largely unaltered. Among them, bit-flips that trap the cores in a hanged state are the majority, while other unrecoverable error sources are less prominent than in CS. In both benchmarks, the error profiles offer vast opportunities for the application of relaxed-reliability error management schemes, dependent on the failure criticality, as opposed to SoA error correction strategies that protect each memory location equally.

To get a more accurate picture of the effect of memory bit-flips on the application output, it is necessary to take into account the number of times that memory location is read/written to. Since distinct memory locations have different access frequencies, statistics on the access count of each memory word in the IM and DM were collected. The errors that the introduced bit-flips result in were then weighted considering the access counts to provide a realistic error distribution. In this way the statistics on the likely result of a bit-flip at runtime could be computed more fairly. To this end, first the applications were simulated under error-free conditions and software counters were employed to extract the count of each IM or DM words read/written to by the respective application. Then, the outputs of the injected errors in each simulation were weighted with the number of times the corresponding bit was read from/written to. Finally, the distribution of the three types of errors was recomputed considering the weighted results, which is summarized in Figure 2.28. Experimental evidence shows a high prevalence of unrecoverable and (especially) SDC errors with respect to masked ones.

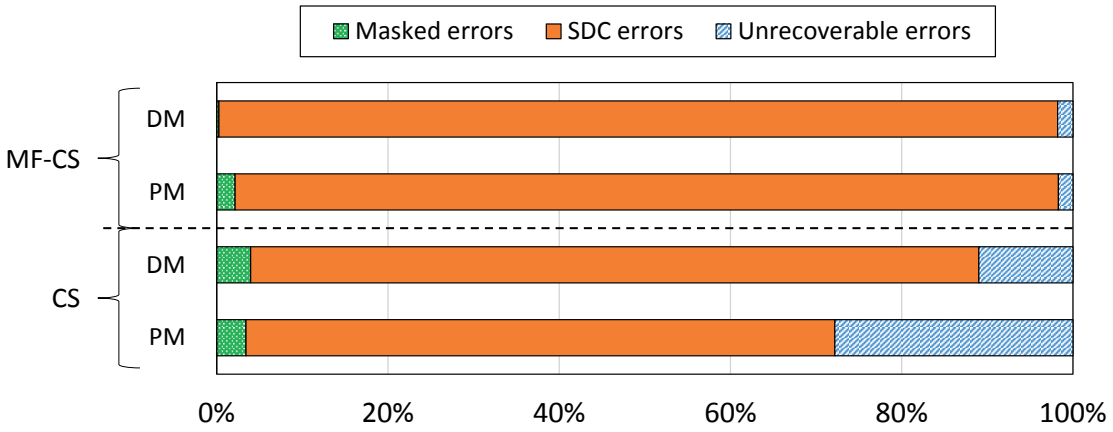


Figure 2.28 – Runtime error profile, considering the access frequency of memory words.

In particular, for CS, while the ratio of unrecoverable errors is non-negligible, the impact of bit-flips causing masked errors is reduced when the number of memory accesses is considered. MF-CS, instead, presents a diminished ratio for both masked and unrecoverable errors, while the impact of SDC errors is even more dominant than in the unweighted distribution.

Evaluation of the error-recovery strategies

A further round of experiments were conducted with forced bit-flips in the same positions as in the previous section, with the execution monitoring enabled. This approach permits the investigation of the effectiveness of the error-monitoring strategies proposed in Section 2.3.3.2. Obtained results show that all unrecoverable errors are countered by the EM, resulting in orderly resets with no side effects after the resumption of execution. This is vital, since any fatal system crash or propagation of errors from a previous computation can be critical to WBSN operation. In addition, the platform is able to detect 96.4 % and 91.8 % of SDC errors for CS and MF-CS applications, respectively, by checking the state of a few critical system elements (synchronization events, FIFO states, IM and DM accesses). Finally, all the masked errors, which do no alter system execution or output were undetected, as expected. These results thereby prove that the proposed strategies are highly effective in countering the SRAM bit-flips.

This study also covers the situation where there may be multiple bit-flips in the memories, each of which might lead to potential application-visible errors. As the EM is able to detect all unrecoverable errors, it can always identify the first bit-flip that leads to a crash. It consequently resets the system, rendering the following bit-flips benign and ensuring that the system is never stuck in a fatal state. Although initial bit-flip(s) resulting in SDC error(s) can avoid detection, eventually the EM detects an unrecoverable or SDC error and resets the system.

Furthermore, by using the receiver-side classification strategy (cf. Section 2.3.3.2), it is possible

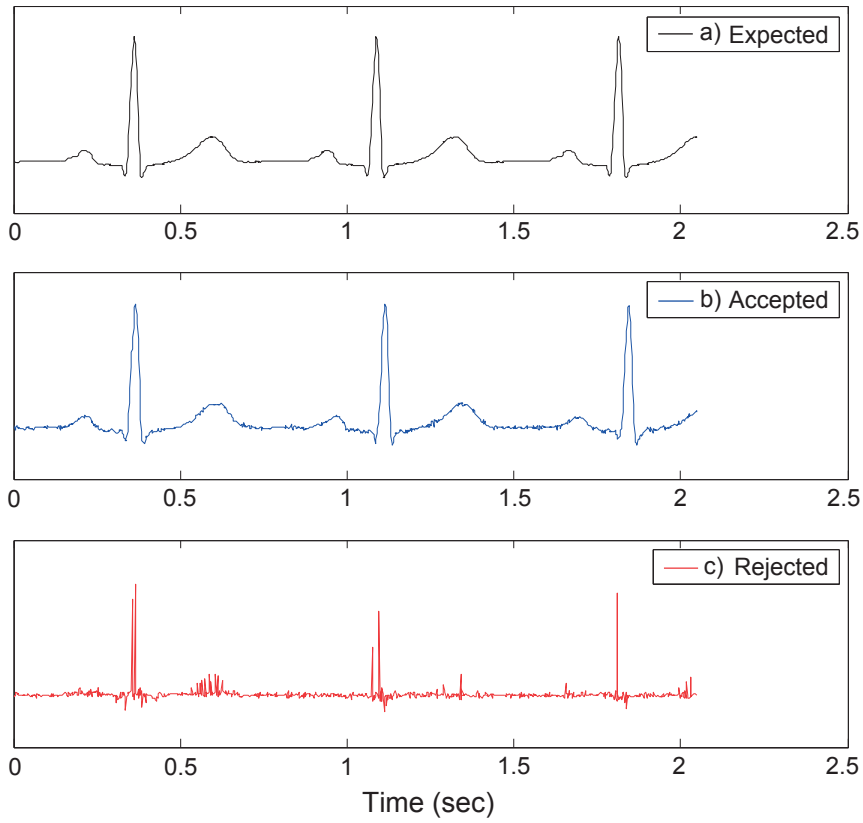


Figure 2.29 – Example of signals impacted by SDC errors relative to an error-free ECG: a) the expected ECG signal, b) example of corrupted ECG accepted at the receiver side, c) example of a rejected ECG window.

to further filter out the SDC errors that were not detected on-board. In particular, highly corrupted signals (Figure 2.29.c) were rejected by the classification strategy while, on the other hand, the signals that were accepted despite SDC errors (e.g., Figure 2.29.b) show a high correlation with the expected ECG (Figure 2.29.a). These data integrity checks were able to counter 82.4 % of the SDC errors which escaped detection by the on-board EM.

Error monitoring strategy overheads

The Execution Monitor [100] comprises various hardware components. In particular, a single register bank tracks when a processor enters or exits a synchronization point, while a stack for each core tracks the synchronization instructions being issued for their legality. Moreover, hardware counters monitor FIFOs for possible overflows, and comparators ensure that instruction and data addresses fall within valid IM/DM ranges. Post-synthesis data (shown in Table 2.2) highlight that the silicon real estate required to implement these architectural elements (indicated in the penultimate row) results in a very limited overhead of 1.2 % of the total system area. To put this into perspective, this overhead is comparable to the one required

Table 2.2 – Area of the multi-core platform and of its architectural components.

Component	Area per element (μm^2)	# of elements
Processor	16969	8
DM bank	69366	16
IM bank	98726	8
Crossbar	17430	3
Synchronizer	6302	1
Execution Monitor	*25 684*	*1*
Total area (μm^2)	2119711	-

by a single crossbar. For comparison, an alternative strategy in which each PM and DM word is protected by ECC would incur a 12x larger area occupation than the EM, even when just the extra storage to accommodate the ECC bits is considered. Moreover, an ECC-based strategy would also require non-negligible energy costs to perform parity calculations on every memory access, which are not required in this strategy.

From the energy consumption point-of-view, the EM presents an overhead of up to 5.7 % of the total system consumption. However, this additional energy cost is eclipsed by the savings that are derived by application of relaxed-reliability computation on the system, as shown in the following section.

Energy/Quality-of-Service trade-offs compared to an exact baseline

The effects of bit-flips at a given supply voltage and droop variance (σ_n) are divided into the three types of errors, considering the distribution ratio showed in Figure 2.28. The detection rates reported in Section 2.3.3.3 are then used to determine the number of erroneous windows that have been discarded. To evaluate the degradation of the system performance, I have defined as a metric for the expected Quality-of-Service (QoS) of the inexact system, the ratio between the accepted correct data and the total processed data. This means that, assuming out of a total of 100 ECG windows processed, if 90 are accepted after error-checks both on-board and on the receiver-side, the QoS would be 90/100 = 0.9 or 90 %. Figure 2.30 shows the QoS degradation depending on the employed supply voltage.

I have pessimistically assumed that any detected error, either by the WBSN or by the receiver, results in the rejection of the full window of samples being processed (i.e., a block of 1024 samples representing ≈ 2 s of the ECG signal). As it can be observed, the presence of bit-flips does not translate into tangible reductions of the QoS until the voltage is reduced close to 1.05 V (considering σ_n = 70 mV). For comparison to an exact system, the supply voltage was set to 1.27 V, that is sufficient to suffer less than one bit-flip per year. A QoS degradation of 10 % therefore allows a supply voltage reduction to 1.02 V (-20.8 %) for the MF-CS benchmark.

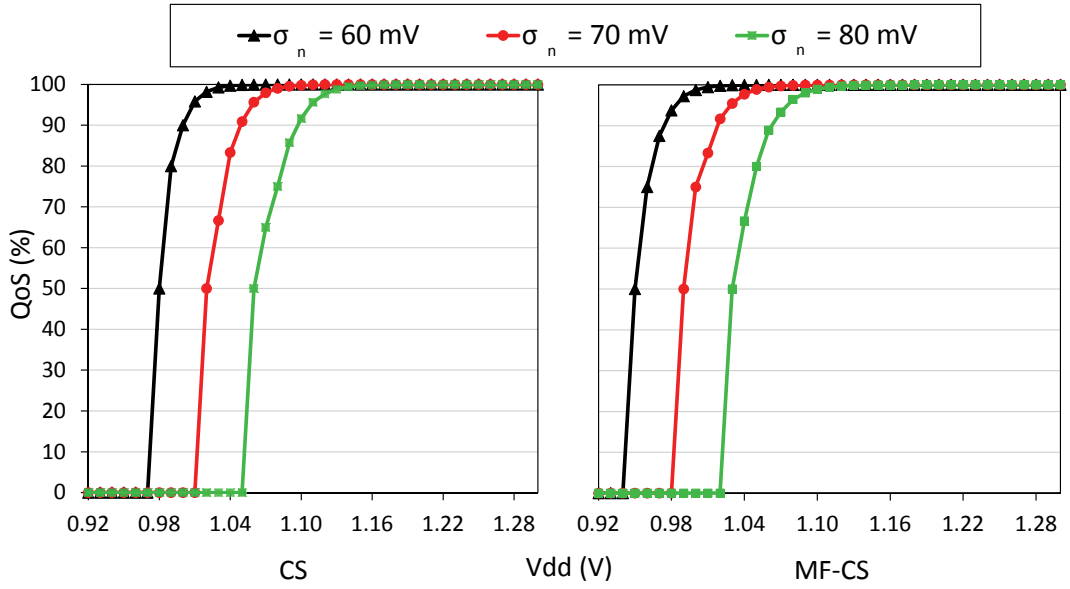


Figure 2.30 – Evolution of the Quality-of-Service (QoS) for different voltage supply values and droop variations.

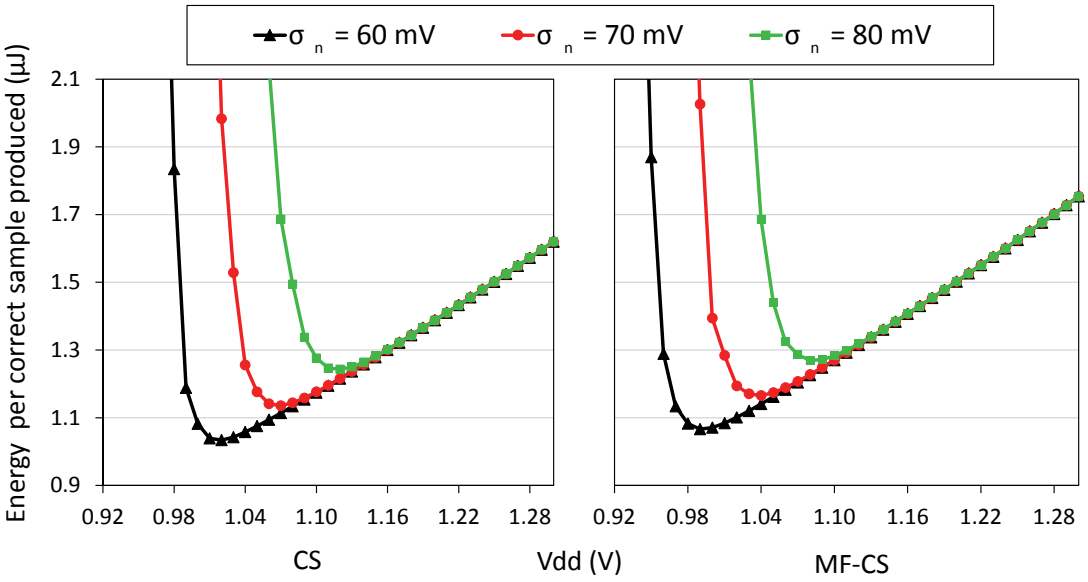


Figure 2.31 – Evolution of the energy consumption per correct output (compressed sample) produced, for different voltage supply values and droop variations.

At this supply level, the platform is able to handle more than 1700 bit-flips per hour and the QoS is guaranteed to be over 90 %.

At lower voltages, the number of bit-flips (and therefore rejected windows) is higher, thereby resulting in a lot of energy used to process data that are finally not usable. On the other hand,

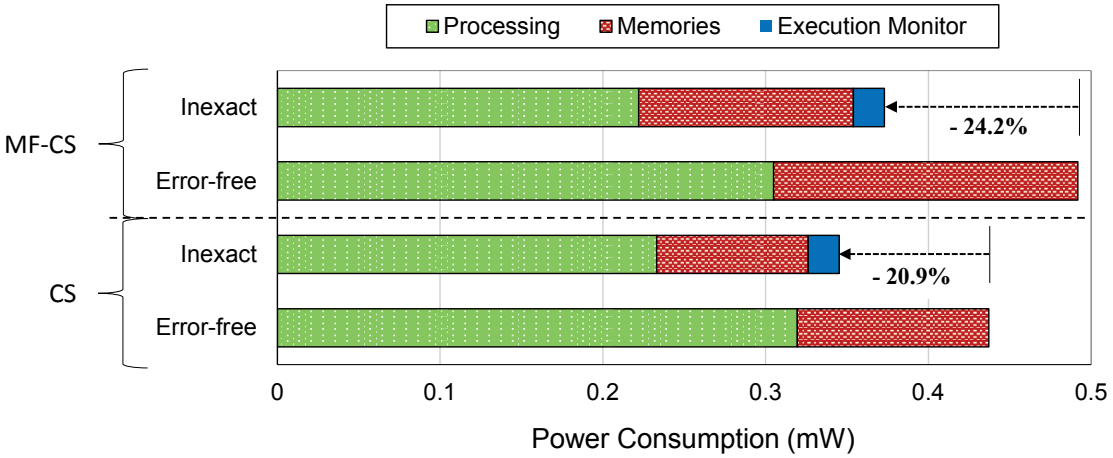


Figure 2.32 – Power consumption comparison of the error-free multi-core system (operating at 1.27 V) with respect to the proposed inexact system (operating at 1.13 V), which guarantees a QoS of over 99 % for both applications.

when higher supply values are used, although the number of correct outputs is much higher, a substantial amount of energy is spent to do the processing. Thus, the main goal of this energy/QoS trade-off study is to assess the optimal point at which the voltage supply of the platform should be set up to guarantee the maximum energy efficiency per correct sample produced. This voltage point also has to ensure that the QoS relative to it is sufficiently high for BSP applications.

Figure 2.31 shows the average energy consumption per valid sample (i.e., samples of windows that have not been discarded either on-board or on the receiver side). As it can be observed, while considering a mean σ_n = 70 mV, the optimal supply point corresponds to a supply of 1.07 V and 1.04 V for CS and MF-CS, bringing power savings of 28.6 % and 32.3 %, respectively. Also, from Figure 2.30 it can be seen that for both cases the QoS degradation is below 3 %, with the corresponding expected error occurrence rates being 240 bit-flips per hour for CS and 900 bit-flips per hour for MF-CS. However, from a medical standard point-of-view, a QoS of more than 99 % needs to be guaranteed, which compels a choice of optimal supply of 1.13 V for the considered applications.

Finally, Figure 2.32 details the overall power consumption of the multi-core system supplied with a voltage level that ensures error-free execution (1.27 V) and the one incorporating the proposed EM supplied with the derived optimal supply voltage (1.13 V). This figure shows that, while the consumption of the EM is not negligible in relative terms (up to 5.7 %), the absolute reduction of the system consumption reaches almost 21 % for CS and up to 24.2 % for the MF-CS application. These savings are derived mainly from voltage over-scaling and hence this study proves that employing ultra-low voltage scaling in association with lightweight schemes for countering ensuing errors results in substantial decrease in the energy budget of WBSNs, while ensuring the production of minimally degraded output.

2.4 Summary of contributions

In this chapter, I have explored the application of voltage over-scaling as an energy-saving strategy in SoA WBSN architectures and the challenges arising as a result. I have concentrated on the failures occurring in WBSN memory subsystems when supply voltages are lowered, since logic elements are comparatively much more resilient to voltage scaling, especially at the considered low operating frequencies. To this end, I have introduced two complementary schemes aimed at ensuring unhindered long-term device operation, even in the presence of memory failures.

First, I have studied the effects on the data-storage buffers at the memory level, showing that a significance-based protection scheme can effectively reduce the energy consumption of WBSN memories. My experiments have shown that by adopting different correctness guaranties in words and bits of varying significance from a BSP perspective, the proposed approach can effectively reduce the energy overhead implicit in data protection, while keeping the output quality degradation within permissible bounds. Experimental results have highlighted that the proposed heterogeneous protection schemes reduce the energy budget of the data memory by approximately 11 %, while tolerating high error rates at reduced supplies, thereby ensuring minimal error in the output.

I have then extended the utilization of voltage over-scaling to the entire WBSN processing system, employing similar aforementioned memory protection schemes. Experimental results have showcased that, by guaranteeing different amount of reliability in the bits and words of varying significance, the energy required for processing the considered PSA application can be reduced beyond the levels attainable by voltage-frequency scaling alone, with a minimal degradation in the output quality. In particular, the proposed scheme reduces the energy budget of the targeted memory buffers by up to 20 % compared to a fully protected one, which in turn translates to overall system-level savings of 5.2 %. Although I have considered the ECG power spectral analysis application, the proposed methodology is applicable in many other real-world BSP applications, as long as they exhibit the same characteristics of processing noisy inputs, generating qualitative outputs and exhibiting a sparse data-representation in intermediate buffers.

In Section 2.3, I have taken an orthogonal stance toward ensuring proper system execution when subjected to ultra-low voltage operations, by introducing a relaxed-reliability computing paradigm with error recovery, for WBSNs. In addition to the previous work, in this study I have included memory failures that occur due to voltage supply fluctuations and fabrication-time variabilities. My experiments have proven that a 100 % correctness guarantee is not required to avoid permanent system-level failures, especially in the BSP domain. Instead, I have shown that high failure rates can indeed be tolerated, while minimally degrading the application-level QoS. As a result, the energy efficiency gains due to aggressive voltage scaling produce more than 24 % overall energy savings at the system level, compared to an equivalent failure-free alternative. I have also shown that by using simple lightweight hardware/software

system monitoring strategies, the platform is able to withstand high error-rates at low supplies. In the process, optimal supply voltage points have been identified for two common BSP applications for which high energy efficiency can be achieved, while at the same time the proposed error-monitoring strategies ensure a very low deterioration of the output quality. My findings establish the benefits of the relaxed-reliability computation paradigm for the design of ultra-low power WBSN architectures, promising to open research avenues aiming at the design of “good enough" systems.

3 Heterogeneous Architectures for Ultra-Low Power WBSN Platforms

3.1 Introduction

Recent studies have shown that, in the near future an enormous number of devices will be connected to the Internet, with an estimated 50 billion or more by 2020 [106]. A majority of these will consist of Internet of Things (IoT)-based autonomous devices, enabling major innovations in the interactions between humans and machines [107]. One of the most important impacts of these devices will be in the healthcare domain, potentially reducing the costs of long-term monitoring of chronic ailments along with improving the quality of life of affected patients [5]. Healthcare monitoring is fast becoming extremely relevant, since the aging of the world population and prevalence of unhealthy lifestyles have made chronic cardiovascular diseases the leading cause of deaths worldwide [2].

In this context, emerging generations of Wireless Body Sensor Nodes (WBSNs) provide a cost effective and unobtrusive solution for performing long-term continuous monitoring of affected patients. WBSNs are able to monitor a wide range of chronic pathologies [6] [108], such as cardiac arrhythmias (from Electrocardiograms, ECG) [109], sleep apneas [110], or Parkinson's disease symptoms [111]. These battery-powered wireless devices are capable of working independently, outside of a hospital environment, with no or little supervision from medical personnel. A key requirement for WBSNs is that they must be highly energy-efficient as they are desired to operate over long intervals with a small on-board battery.

Emerging *smart* WBSNs are able to acquire bio-signals and perform complex Digital Signal Processing (DSP) on-board (also called Bio-Signal Processing, BSP), to extract clinically-relevant features. In this way, they send only a fraction of the acquired data, thereby reducing the amount of information to be transmitted by power hungry communication links. Although such an approach results in increased energy efficiency, these benefits can only be leveraged by performing the BSP stage itself within a small energy envelope.

Thanks to the progresses in the design of domain-specific Analog-to-Digital converters (ADC) [17] [18] and wireless protocols [19], BSP tends to dominate the energy budget of smart

WBSNs [30], so that any increase in its efficiency has a tangible impact at the system level. With the end of Dennard scaling [27], and given the unreliability of traditional SRAM memories as a result of voltage over-scaling, the most promising solution for further increasing the energy efficiency of WBSNs resides in their architectural optimizations to process BSP applications more efficiently, and to reduce leakage consumption. In this context, the major energy concerns in the BSP domain reside in the computation-intensive segments (called *kernels*) inherent in BSP algorithms, and in the ever-increasing leakage component of their power consumption. These issues call for explorations at the hardware and software levels, requiring careful domain-specific solutions. In the work presented in the chapter, I focus on ECG processing, as these signals are the primary indicators of cardiac problems.

3.1.1 State-of-the-art

This section details the existing State-of-the-Art (SoA) of BSP architectures, discussing the advancements and optimizations proposed in the literature, the issues faced by them, and possible solutions.

3.1.1.1 Domain-specific optimizations

In the literature, a number of optimizations have been proposed for WBSNs at the architectural level to achieve high energy efficiency and to sustain the processing of complex BSP algorithms. These methods include acceleration of the execution of BSP applications using additional hardware as well as with multiple processing cores. Dedicated accelerators, like Application Specific Integrated Circuits (ASICs) have been employed to speed up the operation of certain targeted applications, in turn saving computing time and energy. Other solutions include Field Programmable Gate Arrays (FPGAs) that provide wider flexibility in the target application domain. More recently, multi-processor architectures have been successfully employed in BSP, lowering the energy budget. These solutions are further discussed in the following.

Accelerators, FPGAs and general purpose processors

In the context of BSP architectures, various solutions of varying flexibilities have been proposed, aimed at improving performance and/or energy efficiency. A well-known strategy to maximize energy efficiency is to employ dedicated hardware blocks (custom instructions [101] or accelerators [112]) to efficiently process computation-intensive segments of applications (called *kernels*), such as FFT computations, digital filtering and/or matrix multiplications. In this context, ASICs are one of the most widely proposed architectures in the BSP literature for ECG processing, providing high energy efficiency [38] [39] [113] [114]. However, they are also inflexible, as each block can typically perform a single function. These characteristics are even more preeminent when these accelerators are integrated in a multi-core environment [115], because multiple requests for accelerated functions issued by each core must be arbitrated,

thereby limiting the range of applications that can be executed on them [41]. This drawback makes them unsuitable for WBSNs as the latter typically need to cover an extensive range of BSP applications.

On the other hand, FPGAs and general purpose embedded processors have also been probed for BSP architectures [40]. Although FPGAs are highly flexible, thereby being able to execute a wide range of BSP applications, they come at a cost of very high energy and area overheads due to the large amount of hardware and configuration periods that are associated with them. Such high energy overheads (7x - 14x more dynamic power and 5x - 87x more static power than that of a corresponding ASIC [41]) are not admissible for ultra-low power WBSNs. Moreover, although low-power general purpose processors exist (such as the MSP430 System-on-Chip (SoC) from Texas Instruments [116]) that are used in research as well as in commercial platforms [117] [118] [119], they are not optimized for the BSP domain.

Domain-specific low-power processors

Domain Specific Instruction-Set Processors (DSIPs) have recently gained significant research interest [21] [120]. These energy-aware processors, featuring a simplified instruction set and pipeline, are targeted for specific domains such as BSP, but not only for a specific functionality as in ASICs. In addition, they usually present a small area overhead and support advanced power management schemes.

In this context, the TamaRISC [101] is a simplified Reduced Instruction Set Computer (RISC) DSIP architecture aimed at processing BSP applications, whose detailed description can be found in Section 2.3.3.1. It features a highly reduced set of instructions that are sufficient to cover the computations needed in typical BSP algorithms [101]. The processor is a memory-to-memory architecture, with a Harvard-like structure having a three-stage pipeline (fetch, decode and execute stages). The data width is 16 bits and the core comprises 16 registers and 3 external memory ports, for one instruction read, data read and data write each in the same cycle. The register file has 3 read ports and 4 write ports to provide 32-bit double word write-back support.

The TamaRISC comprises 11 unique 24-bit instructions (8 Arithmetic Logic Unit (ALU), 1 general data-move and 2 program flow). It supports addition, subtraction, logical AND, OR, XOR, left/right shifts as well as 16-bit × 16-bit multiplication on signed and unsigned data, producing a 32-bit result. All ALU instructions work on 3 operands, using the exact same addressing mode options for each instruction. Register direct, register indirect (with pre- or post-increment and decrement) as well as register indirect with offset addressing modes are supported. Branching is possible in direct and register indirect modes, as well as with an offset with 13 different condition modes (dependent on the processor status flags: carry, zero, negative and overflow). Using this architecture, the TamaRISC is able to reduce the complexity, and thereby the power consumption, by a huge margin compared to other SoA DSIPs in the BSP domain [101].

Parallel processing

One of the most effective methods to improve energy efficiency in WBSNs is voltage over-scaling. A number of DSIPs are often coupled with advanced power management schemes, able to perform dynamic Voltage-Frequency Scaling (VFS) to adapt to different workload conditions when running BSP routines [29] [42] [43]. However, when supply voltages approach near-threshold values of transistors, they suffer from performance degradation, which limits their ability to work at necessary frequencies, thereby limiting the performance of the associated platform.

On the other hand, BSP applications are generally composed of moderately complex algorithms which manipulate bio-signals coming from multiple inputs. These applications can be optimized to run in real-time (typically embedded) low-power micro-controllers. Moreover, the analysis of multi-input bio-signals involves considerable amount of computation that can be done in parallel. In this context, low-power multi-core parallel processing architectures have been proposed, which increases the parallelism by sharing the computational workload between the multiple processors, thereby improving the platform's performance even under VFS [121] [44]. In order to do so, the DSP algorithms applied to each of the multiple inputs can be parallelized so that the processing of each input signal is done on a different processor core. Furthermore, this approach of using multiple processors also supports software pipelining. In such a scenario, different algorithmic phases are pipelined and each phase is executed on a different core. These approaches aid in reducing the operating frequency of the system by sharing the workload, and hence promise to potentially reduce the supply voltage requirement. In fact, it has been shown, that by using parallel processing it is possible to save up to 66 % energy compared to a single-processor equivalent [23]. Nevertheless, this workload division requires efficient core-to-core notification mechanisms to properly manage producer-consumer relationships, in order to avoid expensive active waiting periods or imprecise periodic polling.

Single Instruction Multiple Data (SIMD) execution

This strategy is derived from sharing processing workloads introduced by the parallel processing of BSP applications [122]. In a typical multiple processor-based architecture, each processor has a designated portion of the instruction memory from where it fetches the instructions to be executed by it. Since identical computational segments in BSP applications can be mapped onto different processing cores, the instructions for the cores doing the same job can be merged together (broadcasted from a single memory segment), when the cores are synchronized. This alleviates the unnecessary accesses by the different cores from their own regions of memory, by streaming multiple requests to the same instruction in a single equivalent. This strategy saves significant amount of memory accesses and hence energy consumption. In fact, crossbar interfaces between memories and processors have been proposed in the literature that support broadcasting [46]. The data processed by each core, however, are potentially different from each other, as they generally execute on bio-signals acquired from different sensors. Thus, in this case, each core is required to be fed with its own set of data.

Code synchronization

In a parallel processing platform, significant energy savings can be obtained by maximizing the SIMD operation mode. Code synchronization strategies for BSP applications are aimed toward that direction. A key to maximize SIMD execution is to ensure that multiple cores execute in lockstep as much as possible, meaning that the cores process the same code segments at the same time. To achieve that, code synchronization approaches typically feature a lightweight hardware/software-based technique: on the software side markers or synchronization points are inserted in the application code to keep track of each processor's execution [47]. This is reciprocated by additional flags in hardware, which are used to store the processors' execution states. A hardware *synchronizer* is used in this situation, which modifies/updates the flags and ensures return to lockstep execution for cores that deviate due to data-dependent branches. This is done by making cores which have already finished a code-segment to wait for the others to resume lockstep execution [47]. Code synchronization technique has been shown to improve the energy efficiency of BSP platforms by up to 50 % [24].

3.1.1.2 Reconfigurable architectures

BSP applications typically consist of moderately complex code that performs arithmetical and logical manipulations on the raw acquired data. These routines generally employ algorithms that contain sections that are computationally very intensive. These code-segments mainly include functions that are executed repetitively over all the incoming samples (e.g., finding the maximum or minimum for each ECG wave while doing filtering). These functions, in turn, embed iterations that form the core of the execution (called *kernels*). In fact, it has been observed that in certain BSP applications, a majority of the computation-time is dominated by these kernels [25], hence pointing to the direction of accelerators that can speed-up their execution, potentially leading to energy benefits.

In this context, reconfigurable architectures are good candidates to take advantage of the energy efficiency typical of dedicated hardware, while providing the degree of flexibility required by a wide range of BSP application kernels. In this scenario, Coarse Grain Reconfigurable Arrays (CGRA) (which typically feature a 2D-mesh of interconnected Processing Elements, PE, with their own control logic and registers) are a promising solution. First, they can be reconfigured at the operation level to process a wide array of applications/kernels, while the cost of their reconfigurability is orders of magnitude lower than that of FPGAs [48] [123]. secondly, they can complement the existing processor-based BSP architectures, thereby sharing the intensive processing load and reducing the execution time of processors.

Existing works in the literature have introduced CGRAs, advocating their use based on their energy efficiency [124], with one being proposed for BSP (Electroencephalography, EEG signals) [125]. In comparison, this work includes a CGRA shared by a multi-core BSP platform, that additionally features support for SIMD execution. Furthermore, while kernel mapping techniques on CGRAs exist in the literature [126] [127], in this research kernel scheduling is

performed aimed at the BSP domain. It differs from previous research as the mapping here is done considering a SIMD CGRA in conjunction with a multi-core WBSN platform.

3.1.1.3 WBSN design using non-volatile memories

Leakage energy is of major concern in emerging WBSNs, with its value increasing as transistors with smaller technology nodes are being conceived. Moreover, since WBSNs spend a significant part of their processing time in the *idle* state while waiting for bio-signals to be acquired, leakage is the dominant component of their power consumption [30]. Traditional SRAM-based WBSN memories constitute a majority of the total leakage power. Being volatile, they cannot be powered-off when WBSN processors are idle, thereby calling for low power-consuming non-volatile alternatives.

SoA WBSNs typically employ flash memories for non-volatile storage, backing up the instruction and data memory contents when the system is switched off, eventually transferring them back to the on-chip SRAM at power-up. However, such systems cannot support fine-grained power-gating over relatively short idle periods to save on leakage, because of two main reasons: first, the real-time deadlines typical of BSP applications cannot be met due to the very long write latencies (i.e., the time required to write a word into the memory) of flash memories ($\approx$ 120 ns) [128] [129]. Secondly, the cost of backing up the full data memory several hundreds of times every second can negate the potential savings obtained from power-gating.

To overcome these challenges, heterogeneous architectures have been proposed in the literature that exploit the benefits from emerging NVM structures (typically Resistive RAM, ReRAM). While a wide range of research exists for emerging NVMs in embedded and high-performance platforms [130] [131] [132] [133] [134] [135] [136], fewer works exist for the ultra-low power BSP domain, which has traditionally been dominated by SRAMs. Furthermore, [137] [138] [139] have proposed 3D integration strategies for combining ReRAMs with traditional embedded memory hierarchies, aimed at enabling faster transfer between the memory levels and for efficient power-gating. The interconnection densities of such memory structures have been proven to be negligible when compared to existing solutions like Through-Silicon Vias (TSVs) [140]. In this work, I focus on ReRAM-based technologies, but similar methods can also be applied to other NVMs such as Spin-Transfer Torque (STT) RAM and flash.

Recent works have shown the emergence of NVMs based on Resistive RAM (ReRAM) technologies, that are able to retain the data stored in them when no voltage is supplied, while at the same time consuming a low amount of power. The authors of [30] have shown that these memories can be integrated with traditional WBSNs, resulting in a hybrid memory subsystem, where the NVM is used as the main storage. However, these emerging NVMs suffer from endurance-related failures, which occur when the memory has been written to for a large number of times [141]. The failure-rates increase as the memory ages along with the number of times that a memory position has been written. These failures generally manifest as bit-flips and are permanent in nature, thereby implying that the concerned failed cell remains

stuck at the flipped value. Although it is possible to revive the cell by applying a higher supply voltage (and thereby current), such a solution is not desirable in ultra-low power processing, as it results in circuit complications and higher energy consumption. An effective method of getting around such failures is wear-leveling, in which memory words with high number of writes are reassigned to locations which have relatively lower wear [142]. In this way, by increasing the time-to-failure, it is possible to increase the lifespan of the ReRAMs.

3.1.2 Contributions and outline of the chapter

This chapter is divided into two broad sections: in the first one, I propose techniques for the optimization of BSP applications for execution on a reconfigurable accelerator. It is achieved by application profiling, selection of computational hot-spots, and finally mapping of kernels on a domain-specific CGRA, that I designed and characterized. Then, I combine the heterogeneous BSP architecture featuring a CGRA accelerator and inexact computation resulting out of memory issues at ultra-low voltage operations, in order to take advantage of the cumulative energy benefits. In this way, I am able to put together my research on relaxed-reliability computation (cf. Chapter 2) and its effects when working on the heterogeneous platform. In the second part, I perform a study of ReRAM endurance in the scope of BSP, studying the trade-offs in the achieved energy efficiency with the incurred overheads required for long-term device usage. The detailed contributions of this chapter are described in the following.

Reconfigurable bio-signal processing architectures

- I propose a heterogeneous CGRA-based reconfigurable architecture for BSP applications. This involves the elaboration of the various CGRA components (number and layout of PEs, memory subsystems, control logic etc.), along with the specification of the interconnections between various PEs.
- I perform an analysis of a wide range of BSP applications to locate the segments of code that are executed the most, thereby selecting the potential kernels that could be accelerated.
- I propose a methodology for the final selection of computational kernels that can be accelerated in the CGRA, by taking into account the various physical and processing restrictions which the CGRA design imposes.
- I derive the execution flow of these kernels, following which I further analyze the chosen kernels for possible optimizations. Then, I map them onto the various PEs of the CGRA, deriving a feasible scheduling of operations.
- I propose a mechanism to orchestrate the execution of BSP applications on heterogeneous resources (multi-core and CGRA), using instruction set extensions to facilitate it, considering constraints on both the hardware and software sides.

- I model failures in the entire memory subsystem of the heterogeneous BSP architecture, considering voltage supply droops and fabrication-time variabilities. In this way, I explore benefits of both CGRA acceleration and voltage over-scaling.

- I employ the error mitigation strategies discussed in Section 2.3.3.2, on the inexact heterogeneous BSP architecture, to evaluate their effectiveness.

- I perform an analysis of the trade-off between the degradation of the Quality-of-Service (QoS) and voltage scaling under the presence of supply fluctuations. In this way I am able to study the energy savings obtained under the various possible conditions (with or without CGRA, and with or without employing inexact computation).

ReRAM-based heterogeneous BSP architectures

- I evaluate the important energy savings achieved with a ReRAM-based heterogeneous memory architecture for predominantly-idle BSP applications, in comparison to traditional SRAM-based platforms, considering various programming voltage ranges.

- I perform an endurance analysis of a heterogeneous ReRAM-based WBSN platform with a real-world BSP application, observing its effects on device lifetime. Furthermore, I quantify the additional memory capacity required for long-term device operation under different error-probability conditions, using a lightweight ReRAM-cell replacement strategy.

- I present the trade-offs between the ensuing output quality degradation and the potential energy benefits, resulting from the power-gating technique used when ReRAMs are involved.

The rest of this chapter is structured as follows. In Section 3.2, I introduce a CGRA-based reconfigurable architecture for the BSP domain that can efficiently accelerate the processing of computational hot-spots. I begin with a description of the details of the envisaged CGRA architecture. Then, I explore the integration of a SIMD CGRA with a traditional processor-based WBSN platform, identifying the potential kernels to be accelerated, before mapping and executing them on the CGRA. I further this study by investigating the application of an interleaved-datapaths CGRA that can parallelize the execution of kernels called from single cores in non-SIMD mode, processing lengthy input data. Then, I explore the combined application of relaxed-reliability computation and CGRA-based acceleration in BSP, aiming to couple the energy benefits derived from both strategies. I investigate emerging ReRAM-based memory architectures and their usage in BSP platforms in Section 3.3, studying endurance-related effects on ReRAMs and proposing a lightweight strategy to maximize its lifetime, which trades-off with additional ReRAM overhead. Finally, in Section 3.4, I discuss the main contributions of the research presented in this chapter and the results obtained.

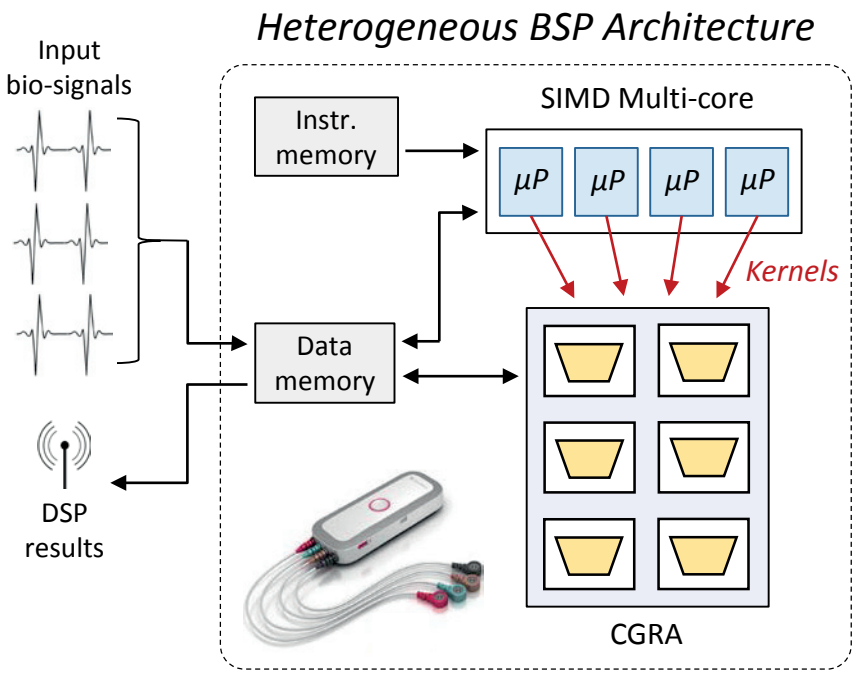


Figure 3.1 – The proposed heterogeneous platform featuring a CGRA accelerator shared by the multi-processors.

3.2 Reconfigurable bio-signal processing architectures

3.2.1 Introduction

The presence of computation-intensive segments (kernels) in most Bio-Signal Processing (BSP) algorithms creates the opportunity of further increasing the energy efficiency of WBSNs, by accelerating the kernels using dedicated architectures. In this context, Coarse Grain Reconfigurable Arrays (CGRA) are a promising solution, as they effectively support the execution of intensive loops, while being flexible and presenting low reconfiguration overheads. In this work, I propose and explore a heterogeneous architecture that features a CGRA accelerator shared between the processors of a SoA multi-core WBSN system [12], as shown in Figure 3.1. Furthermore, I extend SIMD processing to the CGRA, which increases the efficiency when multiple data streams are concurrently processed.

In addition, many BSP applications feature kernels that process typically long input vectors, that are not necessarily called in parallel. To deal with such a scenario, in this work, I also propose a further CGRA architecture that is able to effectively process such kernels by dividing the processing load of inputs into multiple datapaths (collection of logical functional units), thereby potentially further improving performance/energy efficiency. To validate this proposed work, I study a wide range of BSP applications to identify the computational kernels in their algorithms that can be potentially accelerated by the CGRA, performing a detailed study to optimize and schedule them on the CGRA.

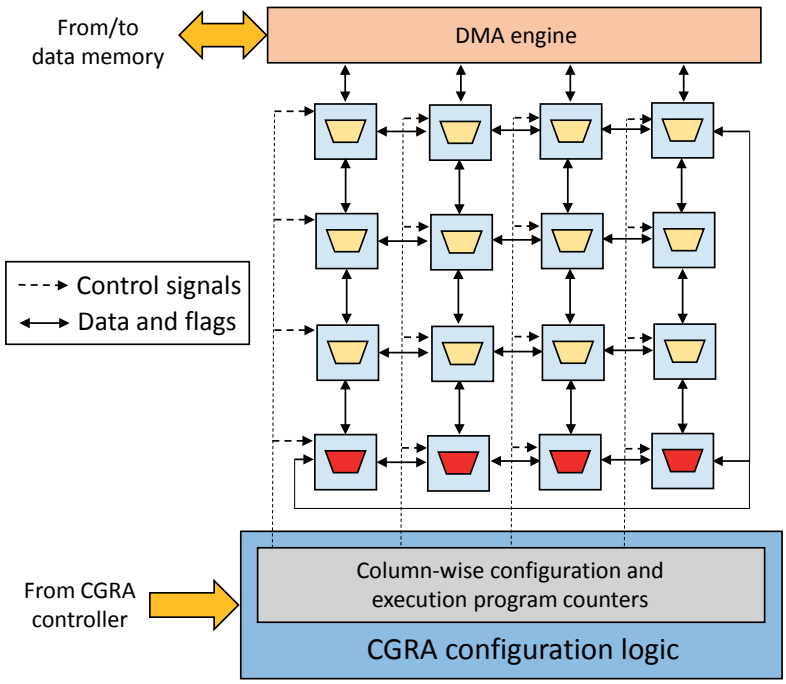


Figure 3.2 – Block scheme of the shared CGRA featuring a single datapath per cell, with one row of the RCs implementing square root calculator (denoted in red). The configuration RAM (also part of the CGRA) is not shown in this figure.

3.2.2 Single Instruction Multiple Data (SIMD) CGRA-based bio-signal processing

This section summarizes SIMD execution on the CGRA, using multiple datapaths. First, a single-datapath-based CGRA is described, from which the SIMD CGRA is derived.

3.2.2.1 Basic single-datapath CGRA structure

CGRA architectures are typically structured as a 2D-mesh of tightly interconnected Reconfigurable Cells (RCs). RCs embed a dedicated ALU coupled with a small local register file. This arrangement allows CGRAs to efficiently execute intensive loops present in BSP algorithms. The configuration overhead of CGRAs, as well as the area devoted to the configuration logic, is orders-of-magnitude smaller than that of fine-grained FPGAs, as only the desired ALU operations and the routing of operands must be specified for each cell. Multiple operations can be cyclically performed, providing a set of configuration words for each RC and activating the proper one during execution [143]. Figure 3.2 provides a high-level view of the proposed CGRA mesh. Each of its cells features a DataPath (DP), which is composed of an ALU, a 4-word register file, and multiplexers able to select input operands (either from the register file, from the ALU output or from the outputs of neighboring cells). The ALU can execute arithmetic and bit-wise operations (AND, OR, XOR, etc.). At the system level, the CGRA is interfaced to the system data memory by means of a multi-channel Direct Memory Access (DMA) block.

This unit uses the memory ports of the processors that requested a given acceleration, and therefore does not require dedicated read and write ports to access values to and from the data memory.

At runtime, each kernel being mapped in the CGRA undergoes a configuration and an execution phase. During configuration, the parameters of the kernel invocation (such as the addresses of inputs and outputs in data memory, and the number of iterations) are received from the issuing processors. They are used to configure the Program Counters (PC) of the employed columns and the required DMA channels. In addition, the bit-streams corresponding to each kernel are also loaded from a configuration memory into the local instruction memories of the PEs. While a single kernel can be configured at a time, execution of different kernels can instead proceed concurrently on separate CGRA columns, effectively employing the available computing resources. During execution, the functionality of RCs is dictated by their active configuration word, selected on a cycle-by-cycle basis by column-wise PCs. After the RCs have finished the computations, the desired outputs are stored by the DMA engine in the system data memory.

Details of the CGRA building blocks

The details of the various building blocks of the CGRA (Figure 3.3) are explained in the following.

- ***Reconfigurable Cells (RC):*** The CGRA mesh consists of 16 processing elements, or RCs, arranged in a 4 × 4 mesh, which is sufficient to map potential BSP kernels. These RCs are interconnected, featuring direct wired connections between neighboring RCs (top, left, right and bottom). In the case when an RC lies at the edge of the CGRA, it is connected to corresponding RC on the opposite side of the mesh in a torus network. For example, if an RC is located in the rightmost column of the CGRA, then it is connected to the RC on the leftmost column on the same row of the mesh (Figure 3.2). As a result, RCs are able to communicate intermediate results and flags produced by them to their immediate neighbors. Moreover, all RCs in the same CGRA column are connected to the Data Memory (DM) via a DMA module by one dedicated port for input and output, respectively. When a processor requests an acceleration, the corresponding data port connection of the core is taken over by the CGRA column(s) that process the acceleration. This is done since the concerned core sleeps during the acceleration execution and does not need to fetch/write anything from the DM while the CGRA processes the corresponding data. This allows a bandwidth of the one data element that can be fetched from/written to the data memory per clock cycle. Finally, each RC in the mesh features a set of configuration registers and a datapath.

- ***CGRA configuration RAM:*** This memory unit is responsible for storing all the instructions corresponding to each and every application kernel meant to be accelerated on

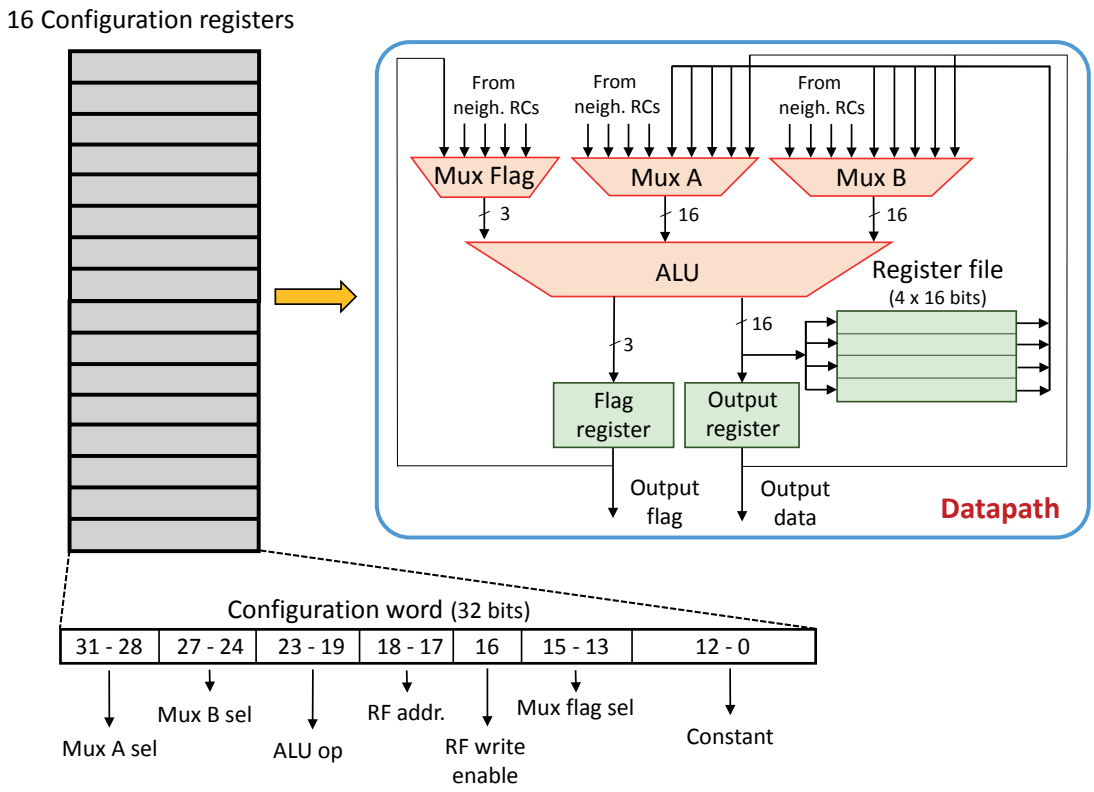


Figure 3.3 – RC architecture of the single-DP CGRA, highlighting the details of the datapath. The CGRA configuration RAM is not shown for clarity.

the CGRA. For this work, the Configuration RAM has a size of 6 kB (1.5 k words of 32 bits), which is sufficient to store the corresponding introductions of all the potential kernels. It employs four read ports, as a result of which it is possible to simultaneously configure all four RCs in a given CGRA column, thereby significantly saving the configuration time.

- ***Configuration Registers (CR):*** These are the local memory of the RCs that are in charge of storing the instructions to be executed by them at individual clock cycles. CRs feature 32-bit long instructions, the decoding of which is shown in Figure 3.3. Each word in this register-file is used for determining the operations to be performed by the CGRA execution components while running a kernel. At configuration time, the instruction words are stored in the CRs of the employed RCs from the CGRA configuration RAM, employing a PC. A similar PC-based strategy is also used during execution phase, to select the proper instruction for a given clock cycle. In the latter case, the PC is extended to support a mapping technique derived from modulo scheduling [144], that is used to map kernels efficiently on the CGRA. This PC is supplied with modulo scheduling parameters, which are then used to select the proper instruction for execution. As a result, this execution PC is able to traverse portions that feature code that are both iterative and non-iterative in nature, which is further explained in details in Section 3.2.2.4.

- ***Datapath (DP):*** Each RC consists of one or multiple DP(s), depending on if SIMD operation mode is supported or not. Each DP, in turn, is responsible for performing the operation desired by the issued instruction from the CR, and has the following components:

  - ***Arithmetic Logic Unit (ALU):*** This unit is the main computational block of the RCs. The ALU receives its two input operands from multiplexers (Mux A and Mux B), and performs the operation indicated by the instruction issued at each clock cycle. It is capable of performing arithmetical operations (addition, subtraction, multiplication, division), logical operations (AND, OR, XOR, NOR, XNOR), and shift operations (left and right bitwise shifts). It produces three flags as a result of each operation, namely the *zero* flag: when the result of the operation is zero, the *negative* flag: when the result is negative, and the *overflow* flag: when the result has more data-bits than what is assigned (standard system data word is unsigned 16 bits). These flags are used in the MUX operation to implement an if-conversion, thus allowing the execution of kernels with conditional statements. In addition, the RCs in one row of the CGRA feature a dedicated square root calculator that is able to compute the square root of a number in a single clock cycle (marked in red in Figure 3.2).
  - ***Multiplexers:*** Each DP features three multiplexers (Mux A, Mux B and Mux Flag). Mux A and Mux B select the values to be fed as the first and second operands of the ALU, respectively. To do so, they receive the output of neighboring RCs, the incoming value from the DM through the DMA, the contents of the register files of its own RC, its own output and the bit-extended constant field of the current instruction, as inputs. The Mux Flag receives the three flag bits produced by the neighboring RCs and itself, and selects one of them to be used in the current ALU operation or passes them on to the neighboring RCs.
  - ***Register File (RF):*** This module serves as the local storage of the RCs for storing temporary data, which might be results from a previous computation, or the value incoming from the DM. It consists of four registers, each of 16 bits, one of which can be written to with the output of the ALU and read from at each clock cycle.
  - ***Output and flag registers:*** The output register serves as a temporary storage, capable of retaining the output produced by the ALU till the next clock edge. At every clock cycle, the corresponding output is thus broadcast to the neighboring RCs. Similarly, the flag register stores the flags generated by the ALU till the next clock edge, when it is broadcast to the neighboring RCs.
  - ***Instruction decoder:*** Each DP also features a module that decodes the incoming instruction from the configuration memory of the RC, to derive the various control bits intended for the different modules inside the DP. In this process, the decoder feeds the multiplexers with the appropriate select signals, the ALU with the operation selection, the register file with the register to be selected and the write enable

signal. It also extracts the constant field of the instruction and extends it to a 16-bit unsigned number. The details of the bits are shown in Figure 3.3.

- ***Controller:*** The CGRA embeds a control logic block that is responsible for the coordination of the different acceleration requests. It is programmed to accept the request with the lowest kernel ID when there are multiple requests at the same time. Upon acceptance, the controller issues an acknowledge signal to the issuing core, so that it can go to the sleep mode (clock-gated). When an acceleration is accepted, first of all the controller retrieves three preamble words unique to the kernel, from its starting address in the CGRA configuration RAM. These words specify the columns to be used and the modulo scheduling parameters that are used in the execution phase.

  The configuration phase is next: first it reads from the DM five words, which specify the starting addresses and length of the input array to be processed, and starting address and length of the the output array to be written in the DM. After this step, the controller configures the columns that are used by the accepted acceleration. In order to do so, it generates the addresses from which the instructions are to be read from the CGRA configuration RAM. Since there are four ports in the RAM, it generates four addresses separated by an offset which is equal to the number of CRs in each RC. The instructions read are then stored in the CRs of each RCs of the employed column(s).

  Once all the employed columns have been configured, the controller enters the execution phase, where all used columns for a given acceleration start executing simultaneously. Execution PCs are used for each column, which are fed with the modulo scheduling parameters that were retrieved during the configuration phase. Upon conclusion of execution, the controller raises a signal that informs the issuing core(s) that their corresponding accelerations have finished processing, so that they can now retrieve the results from the designated area of the DM and resume processing.

- ***Direct Memory Access (DMA) module:*** This block is at the periphery of the reconfigurable mesh and acts as an interface between the CGRA and the DM. The multi-channel DMA is in charge of transferring data corresponding to the core that has issued an acceleration, by using the connection of data read/write ports from the issuing core. Although this approach restricts the bandwidth of data transfer, limiting it to one data word per kernel per clock cycle, it avoids the need of dedicated read/write connections toward the DM.

3.2.2.2 CGRA with SIMD processing

This version of the CGRA (shown in Figure 3.4) has been developed to support accelerations in the SIMD mode. The rationale behind such an approach is that in BSP applications running on a multi-core platform, most of the kernels are called simultaneously by multiple cores performing the same code segment [25]. Although works in the same direction have been presented in the literature such as in [145], it has the limiting assumption that the reconfig-

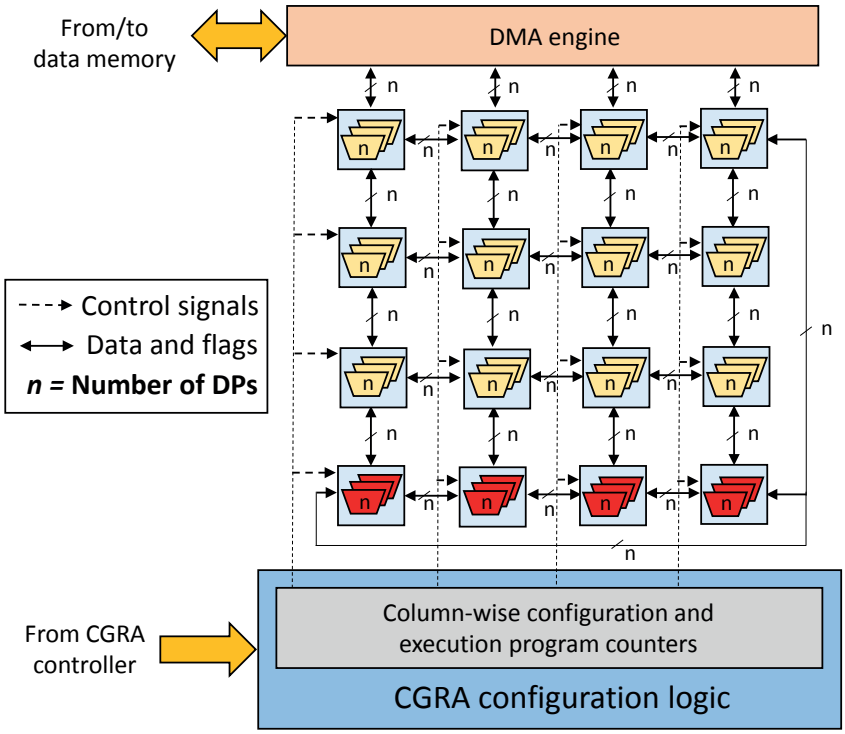


Figure 3.4 – Block scheme of the SIMD CGRA, with one row of the PEs implementing square root calculator (denoted in red). Outputs and flags from each DP are routed to the corresponding DPs in the neighboring RCs.

urable fabric can be accessed only by a single core at a time. My work, conversely, supports acceleration requests by multiple cores, either in a Multiple Instruction Multiple Data (MIMD) or in SIMD mode. This work supports SIMD execution also in the CGRA along with the multi-processors, by adopting a multi-datapath CGRA. In this version, the defining factor is that there are multiple instances of the datapath inside a RC, with each DP intended to execute the acceleration call from a different issuing core of the same kernel (Figure 3.5).

Since different DPs performing the same acceleration require the same instruction to be fed to them, the control logic of the RC is shared by the different DPs, thereby reducing the ratio of the overheads imposed by it. The data to be processed, however, is different for each DP as they are related specifically to the issuing core, and are fed through the multi-channel DMA. In such a scenario, the CGRA borrows the connection between the DM and the issuing processor(s), as the corresponding core remains idle during the whole acceleration. Similar to the single-DP CGRA case, each kernel undergoes a configuration phase and an execution phase. In the configuration phase, the instructions are loaded from the CGRA configuration RAM to the RCs in the columns that are employed. A similar columnar granularity is also chosen while execution, so that the same instruction is fed to all the DPs inside an RC in an employed column. Data sharing between DPs of the same RC or DPs in neighboring RCs other than the corresponding one is not supported, as each set of DPs is specific to a core for a given

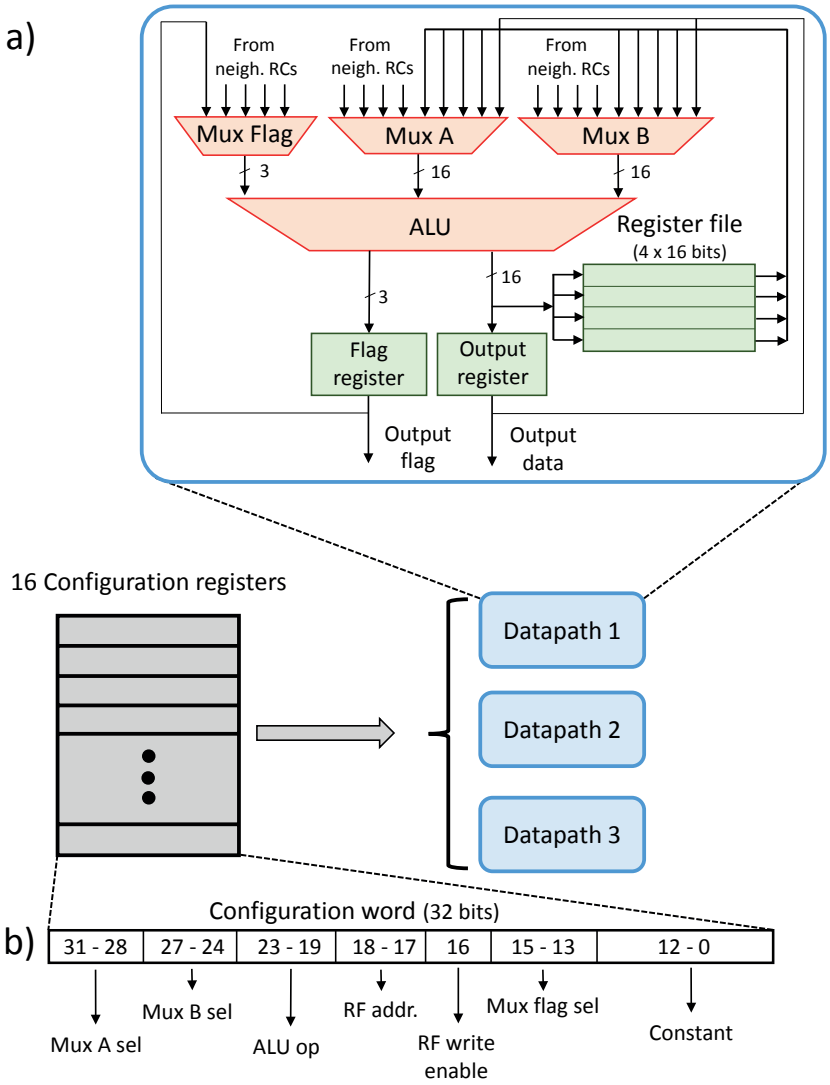


Figure 3.5 – RC architecture of the SIMD CGRA, highlighting a) the datapath structure, and b) the format of the configuration words. All DPs in the RC receive the same configuration word at each clock cycle.

acceleration request and is intended to process a different batch of data. Thus, the output and flags produced by a DP in a given RC is connected to the corresponding DP in the neighboring RCs.

3.2.2.3 Reconfigurable WBSN with SIMD CGRA

The CGRA accelerator is time- and resource-shared by a SoA multi-core system for BSP. The target platform, shown in Figure 3.6, features eight TamaRISC processors, each adopting a Harvard architecture with a three-stage pipeline [101]. They are interconnected to multi-banked instruction and data memories, consisting of 8 and 16 banks, respectively, through

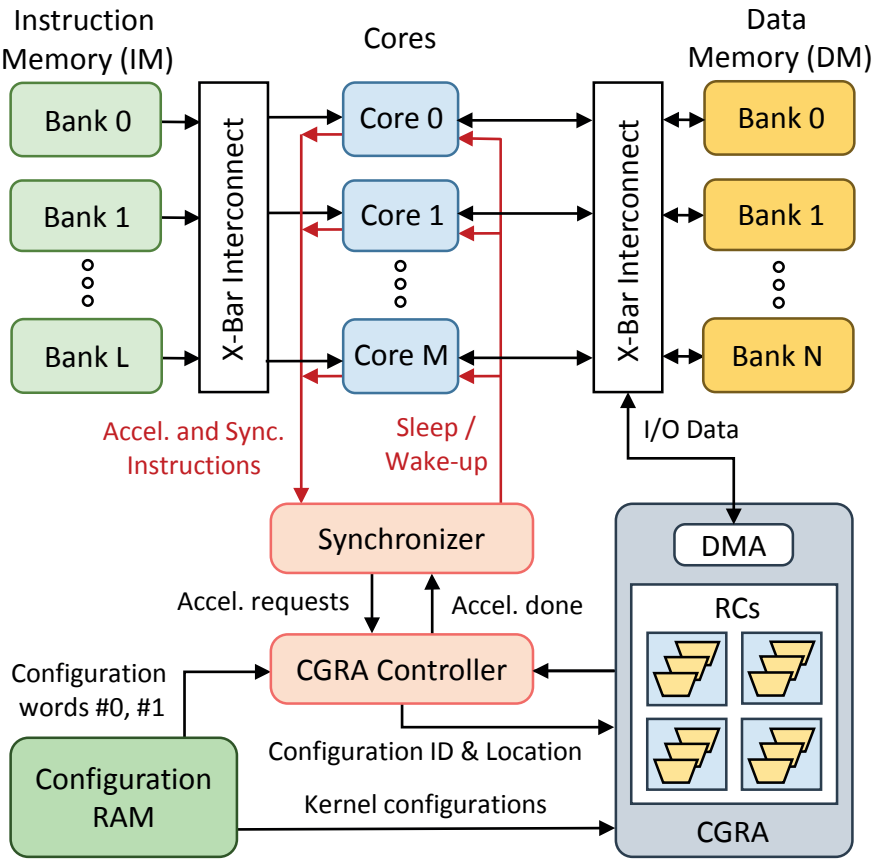


Figure 3.6 – High-level view of the proposed bio-signal processing platform.

combinational crossbars. Each processor can be individually clock-gated when idle (i.e., while waiting for another core to finish, or for a requested acceleration on the CGRA), thereby reducing the contribution of their dynamic energy.

The DM space is partitioned in a shared section, devoted to inter-processor communication, and in private sections, specific to each core. The employed crossbars are able to coalesce memory requests when the same data or instruction is read by multiple cores. A *synchronizer* unit is embedded in the platform that overlooks the execution of the various system components and is responsible for clock-gating the processors. In addition, the synchronizer block also tracks data-dependent branches and runtime divergences occurring as a result, thereby resuming lockstep execution of processors when possible. Moreover, it also coordinates the Instruction Memory (IM) accesses to avoid conflicts when multiple cores request instructions.

The synchronizer is interfaced with the *CGRA Controller*, whose role is to coordinate the acceleration requests from the cores to the CGRA through a request queue. This block checks that enough CGRA resources are available at runtime to map an incoming acceleration request from a core. When a SIMD CGRA is used, the CGRA Controller also coalesces same acceleration requests from different cores. In this case, kernels are concurrently mapped as SIMD-kernels

on different DPs of the same RCs. In case more SIMD requests are issued than the SIMD width of the CGRA, the ones corresponding to the cores with the lower Processor Identifier (PID) are executed first. A CGRA Configuration RAM is employed to store the bit-streams used to program kernels on the CGRA fabric, each word of which dictates the functionality of an RC at a given clock cycle during the execution of a kernel iteration. A further instruction set extension for the processors supports acceleration requests to the CGRA, where a literal specifies the unique kernel ID that the core wants to execute on the CGRA.

3.2.2.4 Kernel mapping on the SIMD CGRA

This part details the methodology employed to select potential kernels to be accelerated on the CGRA and their corresponding scheduling.

Application profiling and kernel selection

The first step for the selection of kernels that can be accelerated on the CGRA is the BSP application profiling using the LLVM compiler [146]. It runs the C version of the given applications and outputs the frequency of execution of each code segment. It parses the various routines and sub-routines employed in the application and the code that is embedded in them. As a result, it is possible to identify the loops that have been executed the most, thereby pointing out the computational hot-spots for the profiled applications. Finally, the code segments are sorted according to the number of times they have been executed. By performing this step, it is possible to have a clear idea of the computationally intensive loop bodies in the applications. An example of the sorted LLVM profiling output is shown as follows:

DFG_rpclass_findmodmax_for.body.126.gv:frequency = 228488
DFG_rpclass_findmodmax_land.rhs.125.gv:frequency = 228488
DFG_rpclass_findmodmax_prefetch.exit135.128.gv:frequency = 228487
DFG_rpclass_findmodmax_for.inc.134.gv:frequency = 221594
DFG_rpclass_findmodmax_for.inc.land.rhs_crit_edge.135.gv:frequency = 216322
DFG_rpclass_findmodmax_if.else74.132.gv:frequency = 142880
DFG_rpclass_findmodmax_if.then61.129.gv:frequency = 85607
DFG_rpclass_findmodmax_land.lhs.true.130.gv:frequency = 50805
DFG_rpclass_findmodmax_if.then.i134.127.gv:frequency = 38602
DFG_rpclass_findmodmax_prefetch.exit.122.gv:frequency = 12210
DFG_rpclass_findmodmax_if.end.123.gv:frequency = 12210
DFG_rpclass_findmodmax_entry.119.gv:frequency = 12210
DFG_rpclass_findmodmax_return.136.gv:frequency = 12209
DFG_rpclass_findmodmax_land.rhs.lr.ph.124.gv:frequency = 12166
DFG_rpclass_findmodmax_if.then82.133.gv:frequency = 4484
DFG_rpclass_findmodmax_if.then69.131.gv:frequency = 2409

This example shows the output of the profiling of the function *findmodmax* inside the *RP_CLASS* application. It shows that the *for loop (126)* has been executed the most number of times, including the statements inside the loop body. The tool returns similar results for other functions and their contents. In this way, it is possible to select the most frequently executed loops that can potentially be accelerated on the CGRA.

Kernel mapping on CGRA

Once the loops with the highest frequency of calls are identified, the next step is to select from them the ones that can be mapped onto the CGRA after conforming to the constraints presented on the hardware and the software sides. As explained before, the CGRA is part of a multi-core processing platform, sharing the DM ports with the cores. When the acceleration request of a core is accepted by the CGRA, the issuing core is clock-gated, while the CGRA executes the required kernel. Upon finishing its task, the CGRA signals the core to wake up and resume execution. For communicating the size and starting addresses of the input data to be processed and the output data to be written by the CGRA, each core has five dedicated registers in the DM that are used by the CGRA.

In addition, upon acceptance of an acceleration request, the CGRA employs the connection between the DM and the requesting core. Since each CGRA column has access to one data port at a time, the kernel mappings need to be done judiciously. If two RCs in a column need different data at the same time, it is not possible to be mapped directly. In this situation, one of them has to wait for a clock cycle to access its data, i.e., till the other RC has finished its access. Hence, the kernel instructions need to be ordered accordingly. In general, the following constraints on the software and hardware sides exist, that are applied to decide if a potential kernel for acceleration can actually be executed on the CGRA or not:

- ***Multi-core platform/application level kernel selection rules***
  - A fixed number of memory-mapped registers per core can be used (read/write) from the software application to configure dynamic parameters of a specific acceleration request (e.g., start address and size of an array of data to read/to write). These registers are embedded in the DM.
  - The search for candidate kernels to accelerate begins first with the highest call frequencies, and then progressively goes down into the list, toward the kernels called less often. The performance of an acceleration request (execution speed) is mainly dependent on:
    * The preparation time of the data (data formatting) before calling the acceleration (e.g., fill-in of the memory-mapped registers of the core requesting the acceleration).
    * The resource allocation time (total amount of RCs on the CGRA, resource access contention with the other cores using the CGRA, number of RCs required

by the kernel acceleration, number of DPs per RC).

* The execution speed of the kernel on the CGRA.

- ***CGRA level kernel selection rules***
  - Only the inner-most loop of each processing block can be considered as a candidate kernel to accelerate. The outer loop can also be selected as a kernel, only if it is possible to unroll the inner loops into an amount of instructions supported by the RCs of the CGRA.
  - The number of RCs considered in this work is 16, arranged in a 4×4 mesh with a torus configuration, with data/flag propagation possible between contiguous RCs. This CGRA size has been found to be sufficient for mapping all of the considered BSP application kernels. Mapping of kernels need to take into account this arrangement.
  - There is only 1 data port per column of 4 RCs. Hence no 2 RCs in the same column can read/write data at the same clock cycle.
  - The prologue, steady-state and epilogues (explained in the following paragraphs) of kernels are determined by 4 bits each, that are needed to address the 16 words present in the CR. As a consequence, the iteration limit for each kernel is determined by 20 bits (since there are a total of 32 bits per instruction), and hence is currently limited to 1048576, which is sufficient for BSP kernels.
  - Dynamic address calculation and DM access from inside the kernel is not supported. Input data (start, length and frequency of access per element in case of an array) has to be determined beforehand.

In this work, I have used an approach to manually map the kernels onto the CGRA, and as a case study, I have explained the mapping of the computational kernel in the Compressed Sensing (CS) application [13]. The following code snippet represents the selected kernel:

```
1  #define INPUT_SIZE      1024
2  #define OUTPUT_SIZE     512    //Compression ratio of 50%
3  #define LFSR_STEPS      9     //LOG2(OUTPUT_SIZE) in theory
4  #define N_ONES          12     //Number of '1's per column
5
6  #define LFSR_SEED     0x6218
7  #define LFSR_POLYNOM      0x3E8F
8  ..........................................................
9  ..........................................................
10 int mask = OUTPUT_SIZE - 1;
11 while (1){
12   //Read ADC
13   sample = mt_sim_readADC(c_id);
14   //Iterate for number of ones in a column
15
```

```
16 //////////////// START OF KERNEL ////////////////
17   for(j=0; j<N_ONES * LFSR_STEPS; j++) {
18     //Add value
19     outputBuffer[lfsr_state & mask] += sample;
20     //Compute next state
21       temp = lfsr_state & LFSR_POLYNOM;
22       temp = temp^(temp >> 8);
23       temp = temp^(temp >> 4);
24       temp = temp^(temp >> 2);
25       temp = temp^(temp >> 1);
26       temp = temp << 15;
27       lfsr_state = (lfsr_state >> 1) | temp;
28     }
29 //////////////// END OF KERNEL ////////////////
30 }
```

This code compresses the input vector of 1024 samples, generating a 50 % compressed output containing 512 elements. A Linear Feedback Shift Register (LFSR) is used to generate a random sparse sensing matrix with few non-zero components (*N_ONES*), thereby leading to a low-complexity implementation, while still making good signal reconstruction possible [13]. This kernel shows data dependencies between successive iterations, and also between the steps inside a given iteration. This condition does not permit starting a successive iteration (benefiting from modulo scheduling) while a current one is still executing, thereby restricting the mapping to a single RC. However, other kernels representing a more complex structure and without data dependencies can be mapped to multiple RCs, spanning multiple CGRA columns, thereby taking more advantage of modulo scheduling.

The cycle-wise schematic diagram for mapping of the *CS* kernel is shown in Figure 3.7. The entire mapping has been divided into two parts for clarity. The scheduling of the kernel on the CGRA RCs is done by employing a technique derived from modulo scheduling [144]. This method includes three distinct stages:

- *Prologue:* This is the part of the code that precedes the iterative segment. Most of the constants and variables that are used in the successive stages are typically loaded in the prologue.

- *Iteration:* This part constitutes the iterative segment of the code (called *steady state* for modulo scheduling). Being typically repetitive over a number of cycles, this segment generally contains the main computational workload. The fixed period between the start of successive iterations is called the *Initiation Interval (II)*. The objective of modulo scheduling is to engineer a schedule for one iteration of the loop such that when this same schedule is repeated at intervals of II cycles, no intra- or inter-iteration dependence is violated, and no resource usage conflict arises between operations of either the same or distinct iterations [147]. Assuming that a loop has n repetitions, if there are m cycles needed per iteration, then the total time spent on the iteration stage is $n \times m$ cycles.

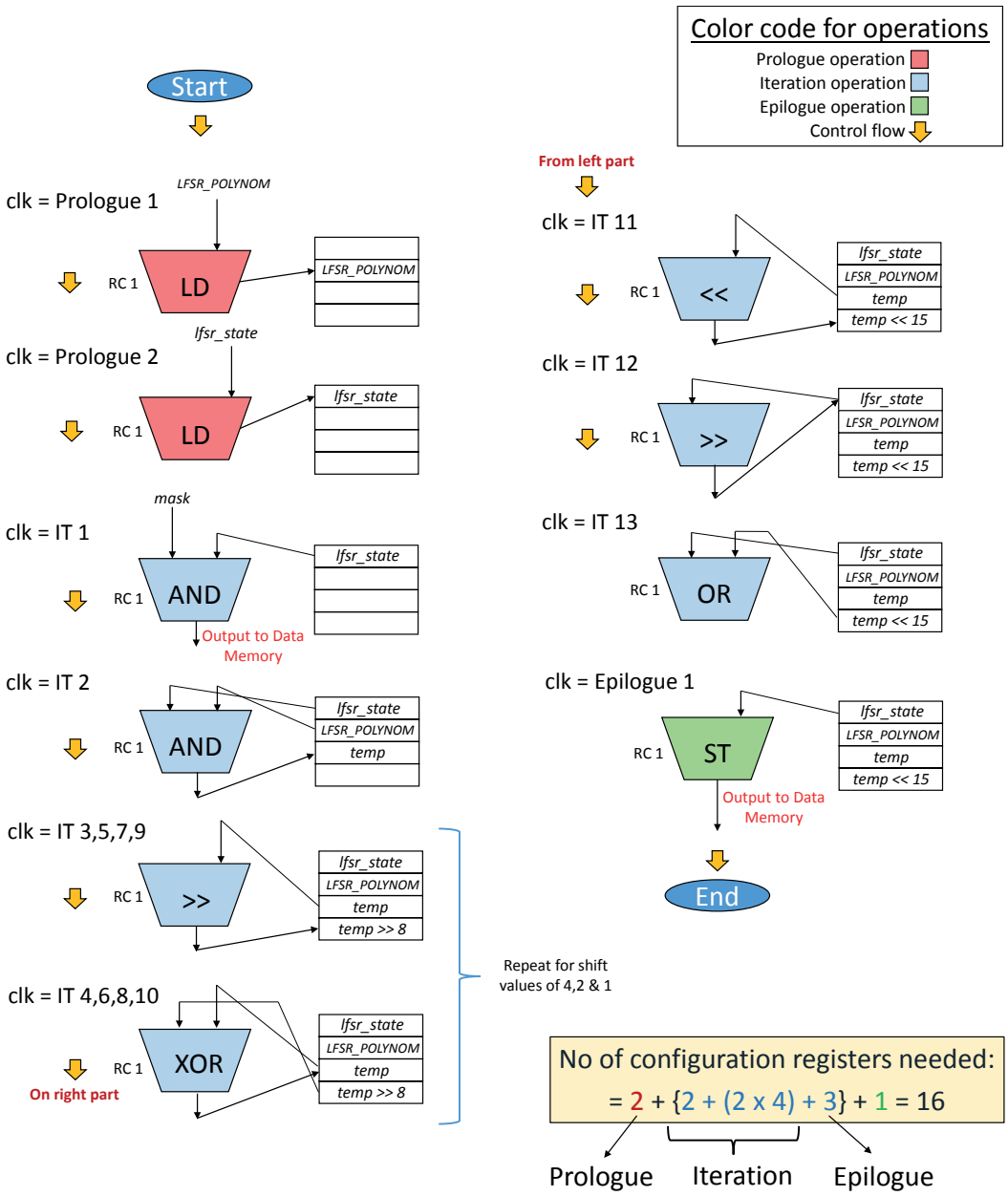


Figure 3.7 – Detailed example of mapping of the *CS* kernel on the CGRA.

- *Epilogue:* This is the part of the code that follows the iteration segment. Once the loop computations are done, the program flow arrives to the epilogue, where computed results are generally stored back to the data memory.

From the kernel code fragment shown as an example before, it can be seen that the value of the constant *LFSR_POLYNOM* is required to be loaded before commencing the iteration step. Also, the initial value of *lfsr_state* is required at this point. Hence, these two values are loaded in the internal RF of a reconfigurable cell (named *RC1* in this example), by employing

two prologue clock cycles. Following this step, the iteration segment can commence, with the corresponding value of *mask* being fed to RC1 at the first cycle of every iteration (Figure 3.7), where it is AND-ed with the *lfsr_state* and sent back to the DM. In the following cycle of the iteration (clk = IT 2), the *lfsr_state* and *LFSR_POLYNOM* are *AND*-ed to formulate the first value of *temp*, which is then stored in a free register in the RF of RC1. In cycle IT 3, *temp* is right-shifted by 8 bits, and this value is stored in the RF. It is then *XOR*-ed to the initial *temp* and the value of *temp* is updated with the new one. This process is repeated with shift values of 4,2 and 1, in cycles IT 5, 7, 9 and XOR-ed in cycles IT 6, 8 and 10, respectively. In cycle IT 11, the current value of *temp* is left-shifted by 15 bits and stored in the RF, replacing its last shifted value which has already been used and is not required anymore in the current loop. At cycle IT 12, *lfsr_state* is right-shifted by a bit and its value is updated with the newly computed one. Finally, in the last step of the iteration segment, the updated values of *lfsr_state* and *temp* are *OR*-ed to obtain the next value of *lfsr_state*. These steps are repeated over *N_ONES* × *LFSR_STEPS*, i.e., 9 × 12 = 108 times, with each of these iteration stages consuming 13 clock cycles. Finally, the epilogue stage takes only one clock cycle, which is used to write back the computed value of *lfsr_state* to the memory. As a result, the number of instructions needed to achieve this kernel scheduling (therefore number of registers needed in the RC's local configuration register) is 2 for prologue, 13 for iteration and 1 for epilogue, which amounts to 16 (Figure 3.7).

Kernel execution

The CGRA controller manages acceleration requests issued by the different cores, using dedicated instructions. Each request comes with the IDs of the kernel and core that has called it, information about the number of columns required to execute it, and the addresses of the positions of the DM from where the input is to be fetched and where the output should be stored after the execution. Upon receiving a request, first the CGRA controller checks if there are enough RC columns free in the CGRA at the time of its arrival to determine if the acceleration is possible. If more than one core requests the same acceleration, the controller groups them to be mapped in a SIMD manner onto multiple DPs in a column. When cores with different acceleration requests compete for CGRA resources, the one(s) with the smallest PID(s) is/are accepted. Once a request is accepted, the controller sends an acknowledgment signal to the corresponding core to inform it to go to sleep mode.

Next, the configuration phase for the kernel begins, where bit-streams of the instructions for the accepted kernel are loaded onto the CRs of the employed RCs, using a configuration program counter. If the same acceleration follows a previously mapped one in the same used RC column(s), then the instructions for the last configuration step are reused, thereby optimizing the process. These bit-streams were priorly derived manually from each of the kernel mappings and stored in the positions of the CGRA configuration RAM corresponding to them. Automation of this process is possible; although it is out of the scope of this thesis, it has been considered for future work. Four read ports have been considered in the implementation

of the CGRA configuration RAM, with each port feeding one of the RCs in a column, thereby parallelizing the process. When an RC is unused, it is loaded with No-Operation (NOP) instructions. The controller is also fed with some preamble words (just before the programming of RCs start) containing information about the employed columns, length of kernel configuration and related modulo scheduling parameters for its execution. The RCs in the corresponding columns are then programmed with the incoming bit-streams. It also informs the DMA about the ports of the DM where to fetch the input information from and where to write the output for each used DP, and the II of the scheduled loop.

Once the programming of all the used columns is done, the execution phase commences. In this segment, an execution program counter is used, which has been fed with the modulo scheduling information by the controller (lengths of prologue and epilogue, iteration interval, number of iterations). It therefore traverses the prologue phase, bringing in the constants or input data needed before the iteration phase starts. Then it iterates over the main loop of the kernel, processing a new data element from the input vector at each new iteration. Finally, it executes the epilogue phase, where the output produced is typically sent back to the DM for further processing by the cores.

Although the CS kernel can be accommodated using one RC in a single column, some other considered kernels require multiple columns, with inter RC data and flag transfers. All the selected kernels were tested individually. To do so, I employed a behavioral VHDL model of the CGRA to test the functionality of the derived instructions, with the corresponding data provided as input at the required clock cycle. Finally, the obtained outputs of the kernels were validated against the corresponding one retrieved by running a software C version of the same.

3.2.2.5 Experimental setup and results

This section details the target architecture adopted to validate the proposed reconfigurable platform, and discusses the obtained experimental results.

Framework

The resulting reconfigurable WBSN platform featuring the SIMD CGRA accelerator was designed and implemented in collaboration with *Loris Duch*, also from the Embedded Systems Laboratory-EPFL, who managed the synchronization of the multi-core system with the CGRA, thereby specifying the various inputs that the CGRA needs from the cores, activation of sleep mode for cores, and the parts external to CGRA execution. A hybrid framework, featuring a post-synthesis netlist and a cycle-accurate simulator of the system (Figure 3.8), has been used to evaluate the energy and performance benefits of the heterogeneous architecture. The RTL implementation, cycle-accurate model and compiler of the employed TamaRISC processors were defined using Synopsys ASIP Designer [104]. Three scenarios of SIMD parallelization have been considered for the CGRA, featuring 1, 2 and 3 DPs per RC column, respectively. The

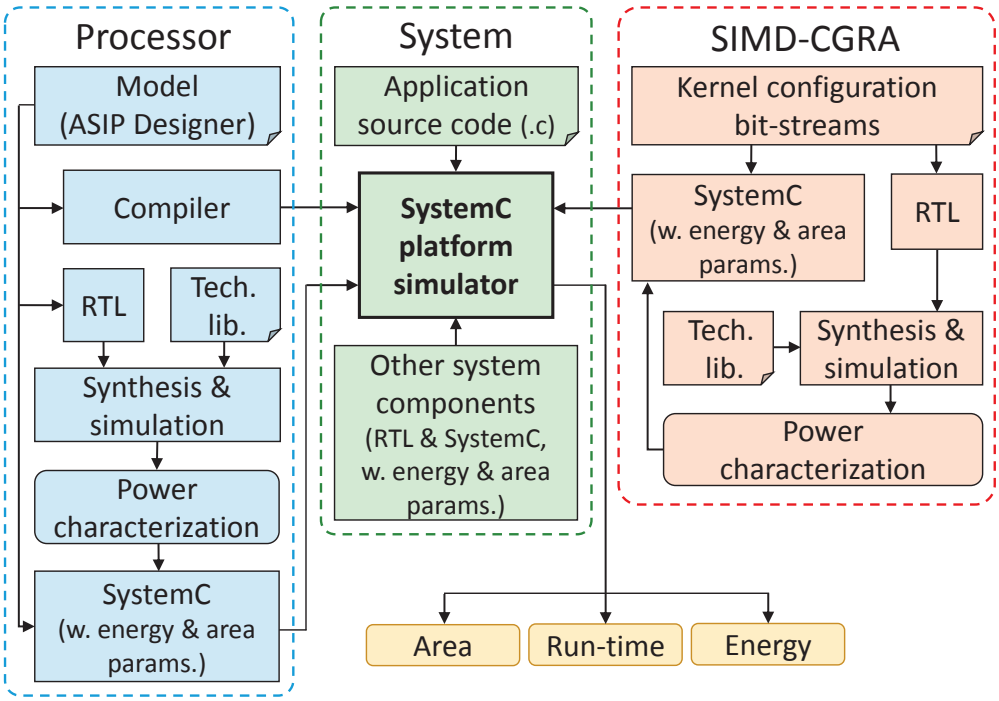


Figure 3.8 – Block scheme of the experimental framework, comprising RTL and cycle-accurate system views.

CGRA has been implemented at the Register-Transfer Level (RTL) using VHDL language, with behavioral testing done employing Mentor Graphics Modelsim [148] and synthesized using Synopsys Design Compiler [149]. The CGRA mesh consists of a 4×4 structure of RCs, which is enough to map all the kernels considered in this work.

The runtime behavior of the SIMD system was implemented as a module in a cycle-accurate SystemC simulator of the entire architecture. The complete platform consists of 8 TamaRISC processors, with an associated Data Memory (DM) of 64 kB (32 k words of 16 bits each) and an Instruction Memory (IM) of 96 kB (32 k words of 24 bits each). The CGRA configuration RAM employed to store the bit-streams corresponding to the kernels has a size of 6 kB, which is sufficient to store the bit-streams of all the considered kernels. The operating frequency of the system is set at 1 MHz, with CS requiring 2 MHz, which meets the real-time constraints. The area and power characterization of the CGRA has been done with a post-synthesis netlist using a 65 nm UMC low-leakage technology library [105] at 0.9 V. In this step, the execution of individual kernels was simulated in the CGRA. The switching activities of the CGRA ports were also taken into account, with a default probability of 50 % assigned to the input data. In order to avoid lengthy hardware simulations, the performance of the multi-processor architecture was evaluated while executing small synthetic benchmarks, with and without SIMD execution. The energy profiles of each computing component (processors, RCs, memories, interconnects) were derived following this analysis.

The processors enter sleep mode (therefore clock-gated) whenever a requested acceleration is accepted by the CGRA, to save energy and since (as opposed to power-gating) the values in their internal registers must be preserved. On the other hand, the CGRA DPs that are unused for a given kernel execution are power-gated, while the configuration memories are clock-gated when idle so that they retain their content, which can be reused if the same kernel is invoked again. The obtained energy profiles, together with the runtime of the considered kernels were then imported in the cycle-accurate simulator written in SystemC, allowing faster experimental evaluations. The simulator outputs execution metrics such as the number of active and inactive clock cycles for each processor and the CGRA, and the number of IM/DM accesses. Next, these parameters are combined with the energy profiles obtained in the previous step to obtain the total energy, area, and performance evaluation of each kernel. Furthermore, the energy spent in each individual kernel call relative to a BSP application can be aggregated to obtain the total energy consumed by the CGRA, while executing that application. This value, in addition with the energy consumed by the multi-processor system (procured from a dedicated power model involving the consumption of the various system components at 65 nm technology with 0.9 V supply), provides the energy consumption of the entire heterogeneous system.

Baselines and proposed systems

To compare the energy and performance of the proposed reconfigurable WBSN platform, three different architectural configurations were considered, as follows:

- ***Multi-core-only architecture:*** This architecture is similar to the one in [47], featuring eight processing cores, but without any CGRA acceleration. The entire processing is done by the cores, with SIMD execution supported to decease the number of accesses to the memories. A comparative evaluation with respect to this baseline therefore explores the efficiency of a CGRA-based WBSN in the BSP domain.
- ***Multi-core architecture with single-DP CGRA:*** In this case, the processors are interfaced to a shared CGRA accelerator featuring only one DP per RC [150]. Hence, the SIMD execution of kernels is not supported in the CGRA, with each request being mapped on different RC columns.
- ***Multi-core architecture with SIMD CGRA:*** This platform is the one proposed in this study, described in Section 3.2.2.3. It features multiple DPs in each RC, permitting the efficient execution of SIMD kernels. In this study, CGRA instances with two and three DPs per RC have been considered. The utilization of the SIMD CGRA is justified for applications where the kernels are mostly called simultaneously by multiple processors (such as multi-lead ECG processing), as the energy-hungry control logic of each RC (which consumes ≈ 2x energy and similar area compared to a DP, cf. Table 3.2) can be shared by multiple DPs in the RC. In contrast, for a single-DP CGRA, control logic needs to be assigned to each DP.

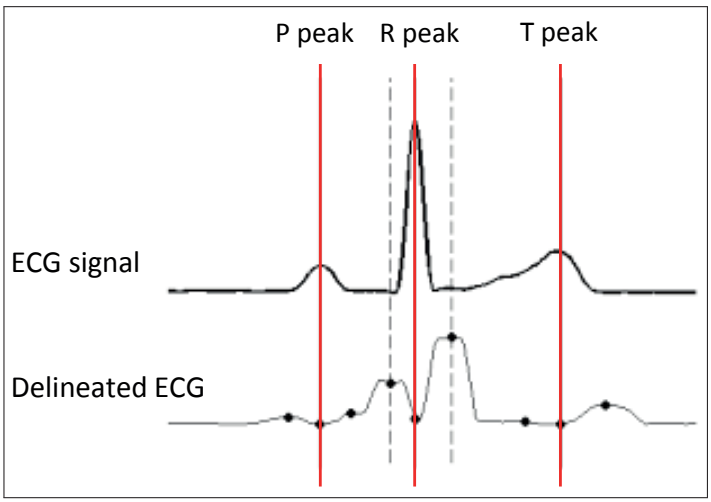


Figure 3.9 – Acquired ECG signal (top) and its MMD-delineated version (bottom).

Target bio-signal processing applications

In order to test the effectiveness of the proposed heterogeneous system, I have used a set of widely employed BSP applications that are listed below. These applications cover scenarios exhibiting fully parallel and independent computation as well as those where producer-consumer relationship is involved between cores. The number of cores required for each application mapping is presented in Table 3.1.

- ***Morphological Filtering (6L-MF):*** This benchmark application features an algorithm which removes artifacts (due to muscular activity, system AC supply interferences and base drift caused by breathing) from an ECG acquisition (for details see Section 2.3.3.3). In order to do so, it uses structuring elements to remove low- or high-frequency noise components, according to the algorithm described in [14]. This benchmark operates in parallel on different input ECG streams and is therefore mapped on different cores. In this work, I have considered a variant of this application which has 6 input ECG leads.

- ***Multi-scale Morphological Derivatives-based delineation (6L-MMD):*** The objective of delineation algorithms is to find the fiducial points of an ECG signal, i.e., the onset, peak and end of the characteristic ECG waves (P, QRS complex and T) (cf. Section 1.1.2). In the case of MMD, a multi-scale morphological derivate of the ECG signal is employed to delineate it (Figure 3.9) [151]. The morphological transformation translates peaks on the input signal to pits on the transformed one, while peaks or sudden changes in slope of the transformed signal highlights onsets or ends of waves in the input one. A search on the MMD transform for a negative value exceeding a dynamically-adjusted threshold retrieves the peak of the R wave; peaks (or sudden changes in slope) around it retrieve the onset and end of the QRS complex. Before and after the found QRS complex, tuples of zero-crossing points mark the presence of the P and T waves respectively. P

Table 3.1 – Kernels utilization per application. Each cell indicates the number of processing cores requesting each acceleration.

Apps / Kernels	6L-MF	6L-MMD	RP-CLASS	8L-CS
Dbl Min Srch	6 cores	6 cores	3 cores	-
Dbl Max Srch	6 cores	6 cores	3 cores	-
Min Max Srch	6 cores	6 cores	3 cores	-
Sqrt 32	-	1 core	1 core	-
Lin Srch	-	1 core	1 core	-
Apply RP	-	-	1 core	-
Lin Min Max	-	-	1 core	-
CS	-	-	-	8 cores

and T peaks are identified by the minimum value in between the crossing points, while their onset and end are identified by the maximum values before and after the crossing points.

This application relies on MF to filter the acquired ECG signals. It subsequently performs a Root-Mean-Square (RMS) combination of the filtered streams, and delineates the fiducial points. It is divided into three different tasks: the MF is done on 6 leads, mapped on the first 6 cores, while the RMS combination is performed by the next core, and the MMD is done by the remaining processing core.

- ***Random Projection-based CLASSification (RP-CLASS):*** This benchmark uses a neuro-fuzzy classifier, operating on a single-lead, to identify pathological heartbeats by applying a random projection over the ECG samples [152]. When a heartbeat abnormality is detected, a 3-lead MF (2 cores, excluding the one used in classification) + MMD (1 core) is activated for a short window of signal. In this way the 4 cores performing MF + MMD are seldom activated (only when an abnormal wave is detected), thereby optimizing the computation and saving energy. Thus, the RP-CLASS application is mapped onto 6 cores, considering 3 ECG input leads, among which the 4 cores in the delineation chain are seldom activated.

- ***Compressed Sensing (8L-CS):*** This application performs a 50 % lossy compression on the input ECG stream (cf. Section 2.3.3.3) [13]. I have employed an 8-lead version of this application with the processing relative to each lead mapped onto a different core.

Kernels accelerated in the CGRA

The kernels to be accelerated in the CGRA have been selected using the methods and constraints described in Section 3.2.2.4. They were then mapped onto the CGRA, adapting an

approach derived from modulo scheduling, defining the number of columns and RCs required, operations to be performed at each clock cycle, number of cycles per iteration and the content of each configuration word. The selected kernels are described in the following [25].

- ***Dbl Min Srch:*** It performs a search of the first and second minimum values inside a window of samples. This kernel is used by the cores running the 6L-MF, 6L-MMD and the delineation part of RP-CLASS.

- ***Dbl Max Srch:*** Similar to *Dbl Min Srch*, this kernel performs a search of the first and second maximum values inside a window of samples. This kernel is also used by the cores running the 6L-MF, 6L-MMD and the delineation part of RP-CLASS.

- ***Min Max Srch:*** It determines the minimum and maximum limit values used during the erosion and dilation steps [14], executed by the cores running the 6L-MF, 6L-MMD and the delineation part of RP-CLASS.

- ***Sqrt 32:*** It executes a 32-bit square root algorithm. This kernel is used by one core performing the RMS combination of the filtered ECG signals (in 6L-MMD) and consequently by one core in RP-CLASS.

- ***Lin Srch:*** It performs a linear search of the two minimum values and two maximum values inside an array of samples. Its execution is more compact than the one when the kernels *Dbl Min Srch* and *Dbl Max Srch* executed sequentially. This kernel is used by one core executing the RMS combination of the filtered ECG signals (in 6L-MMD and RP-CLASS benchmarks).

- ***Apply RP:*** This kernel calculates the random projection [11] on a signal window, by performing a matrix-vector multiplication. This kernel is used by one core executing the RP classifier.

- ***Lin Min Max:*** It performs a linear search of a single minimum value and single maximum value inside an array of samples. This kernel is used by one core performing the classification task.

- ***CS:*** This kernel computes the sequence of random indexes necessary to perform compressed sensing. It is the only kernel used by the cores running the CS algorithm, and constitutes a majority of the application workload.

Table 3.1 shows the kernels that are accelerated for each BSP application and how many cores request them, highlighting the benefits than can be achieved by employing SIMD execution.

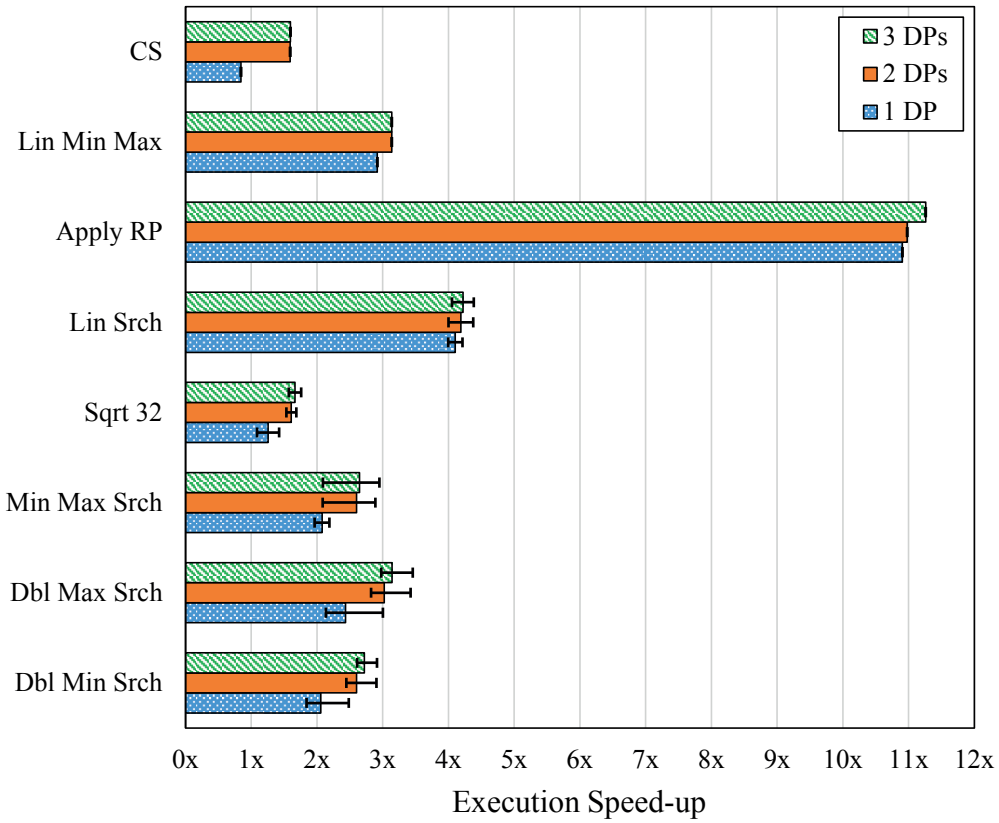


Figure 3.10 – Average speed-up of kernels running on the multi-core + CGRA platform, compared to execution only on the multi-core platform.

Kernel speedup evaluation

Figure 3.10 compares the speedup obtained for the various considered kernels when executed in the different instances of the CGRA, with respect to the time needed to execute them in the processor-only platform. The presented data considers both configuration and execution time, as well as the time required to manage the acceleration requests. The speedups range from 0.8x to 11.3x, depending on the structure of the kernel and the number of DPs engaged in the CGRA (SIMD width). Since the CGRA resources are limited and the kernels compete for them, their speedups vary across calls, with the average, maximum and minimum speedup values reported in Figure 3.10 for the cases having a non-negligible variance. Except for the CS kernel, all others require a smaller execution time on the CGRA, compared to the processors. The CS kernel executing on 1 DP CGRA is an exception (speedup = 0.8x, i.e., slowed down) due to the high amount of contention during its execution, which can be reduced significantly by employing multiple DPs, as it is frequently called by multiple cores in parallel. In such a case, each instance of the CS kernel can be effectively mapped onto a different DP, thereby improving runtime performance. Indeed, a speedup of 1.6x is noted in the case where 2 and 3 DPs are employed in the CGRA; cases which also show substantial speedups compared to a processor-based execution, for all the studied kernels. Although the kernel speedups when

using a SIMD CGRA are mostly similar to those when employing a 1 DP CGRA, nevertheless the former results in significant energy reductions, which is the main objective of the CGRA, as discussed in the following.

Kernel energy consumption

The energy consumed by the various kernels is showcased in Figure 3.11, considering different SIMD widths, normalized to a processor-only (Software, SW) version. The values shown in the figure represent the entire energy consumption of the kernels, aggregated across all invocations in the corresponding benchmark. It can be seen that CGRA execution is more energy-efficient for all the kernels, with respect to a SW-only execution. Moreover, further energy savings can be achieved when a SIMD CGRA is employed. This is even more prominent for kernels that exhibit a high ratio of concurrent calls by multiple cores (*Dbl Min Srch*, *Dbl Max Srch*, *Min Max Srch* in 6L-MF and 6L-MMD, and *CS* in 8L-CS), with ≈ 5000, ≈ 5800, ≈ 28 000 and ≈ 40 000 calls, respectively. However, for kernels that are not called a lot in SIMD mode (such as *Apply RP, Lin Srch* in RP-CLASS), the energy gains obtained by using a SIMD CGRA compared to a 1 DP CGRA are negligible. When a CGRA with SIMD width of 2 DPs is considered, the aggregate savings averaged over all applications where these kernels are used range from 53 % (*Dbl Max Srch*) to more than 90 % (*Min Max Srch*). However, the energy savings achieved by moving to a 3 DP CGRA is limited to only a handful of kernels, with the benefits being relatively very small compared to a 2 DP CGRA.

System-level energy consumption

The energy consumed by the entire multi-core platform with and without the SIMD CGRA is presented in Figure 3.12. The obtained results show that significant energy savings, ranging from 9.2 % to 40.2 %, can be achieved over the SW-only case. These savings are obtained by transferring the computation-intensive workload from the processors to the CGRA. Since the CGRA processes these kernels much faster and more efficiently than the cores (as seen from the previous sections), the system effectively consumes significantly lower amount of energy, even when the overheads are included. Moreover, in case of the SIMD CGRA, the control logic is shared among the DPs, thereby increasing its energy efficiency compared to cores which have their dedicated control logic, while at the same time trading off some flexibility. The maximum reduction in energy consumption is observed in the case of 8L-CS, which is expected as most of the application workload is concentrated in the *CS* kernel. In particular, in this case, using a SIMD CGRA brings about more savings compared to a 1 DP CGRA because of the shared control logic in the former and as 8L-CS is a highly parallel application where the *CS* kernel is called by (mostly) eight cores simultaneously. Thus, for applications where kernels are mostly called in SIMD mode, it is highly beneficial to use a CGRA featuring multiple DPs.

Large gains in energy are also observed for 6L-MF and 6L-MMD applications, which also spend a significant amount of their processing time in the accelerated kernels. However, for

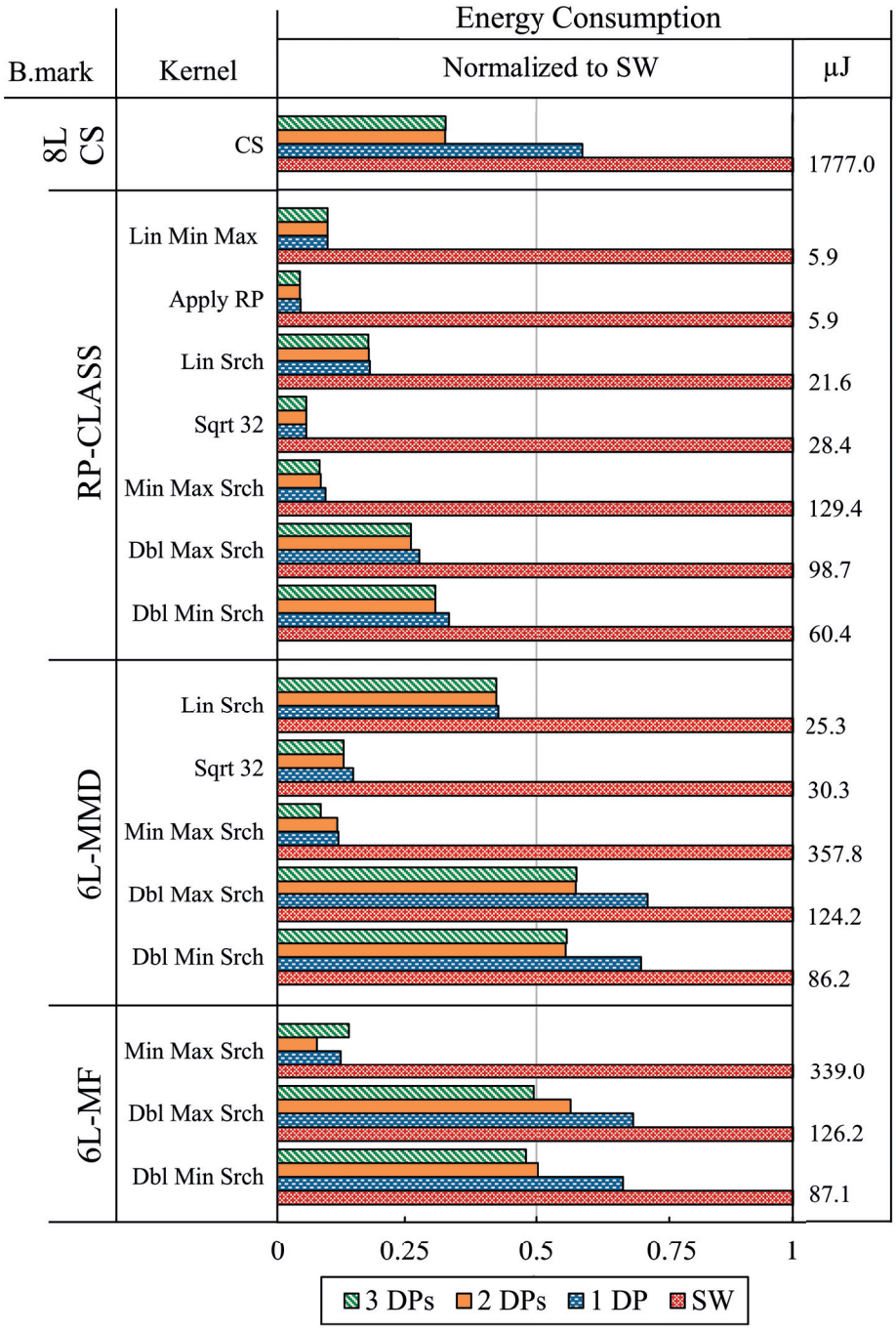


Figure 3.11 – Energy consumption of the different kernels (for the employing application). For each kernel, the bars are normalized to the SW-only consumption, which is indicated on the right side of the graph.

these applications, the reduction in energy overhead is not substantial between 1 DP and SIMD CGRAs, because a majority of their kernels are called in a non-SIMD manner (i.e., at different instances of time by different cores). Even in the case of the RP-CLASS application,

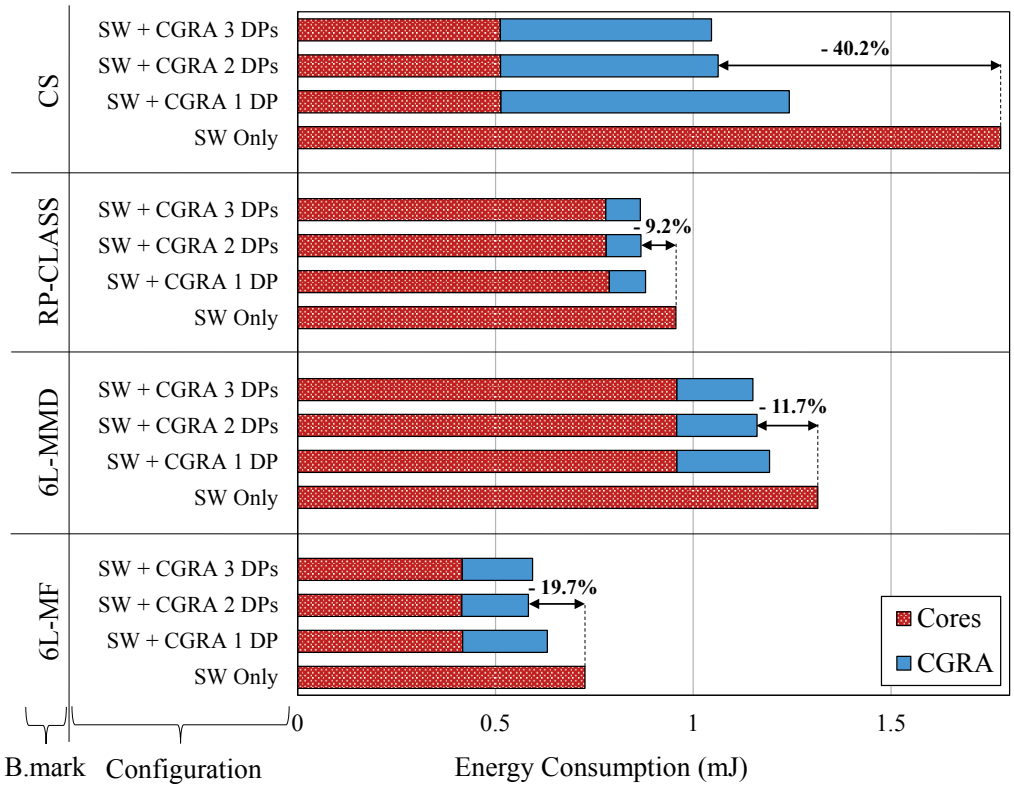


Figure 3.12 – System energy consumption for processing the different benchmark applications.

which has far fewer acceleration calls (≈ 150), significant energy savings of 9.2 % are noted when a SIMD CGRA is used. Again in this case, upon inspection, it can be seen that most of the identical kernel calls happen non-simultaneously, thereby justifying the preceding observation. Finally, Figure 3.12 points out that using the SIMD CGRA improves the energy efficiency in all cases over a processor-only platform (both for applications with high and low number of SIMD kernel calls, except 6L-MF on 3 DPs), as the CGRA takes advantage of parallel processing.

System area exploration

Table 3.2 details the breakdown of the area footprint of the different CGRA components. As it can be seen, the CGRA Configuration RAM accounts for a significant part of the area occupied, amounting to a bit more than half of it in the 2 DP case. The CGRA mesh itself is quite compact, with the CRs and the ALUs being the biggest contributors. The *ALU_sqrt* occupies almost double the area of a simple *ALU* since it has additional circuitry that supports the calculation of the square root of an unsigned number. Each DP inside an RC (ALUs + local register file + multiplexers) is smaller than the configuration memory of the RC itself. Since the CR and the control logic are shared among the different DPs, the overhead of doubling the SIMD width from one to two is therefore ≈ 34 %.

Table 3.2 – CGRA area exploration (in 65 nm UMC technology library).

Component	Area (μm^2)
Configuration register (per RC)	6721.5
CGRA control	5227.9
Cell control (per RC)	881.7
Register file (per DP)	1374.2
Multiplexers (per DP)	822.9
ALU (per DP)	3962.8
ALU_sqrt (per DP)	7859.1
Configuration_RAM	278978.2
Total CGRA area (1 DP)	**361591.2**
Total CGRA area (2 DPs)	**484900.7** (+ 34 %)
Total CGRA area (3 DPs)	**593348.6** (+ 64 %)

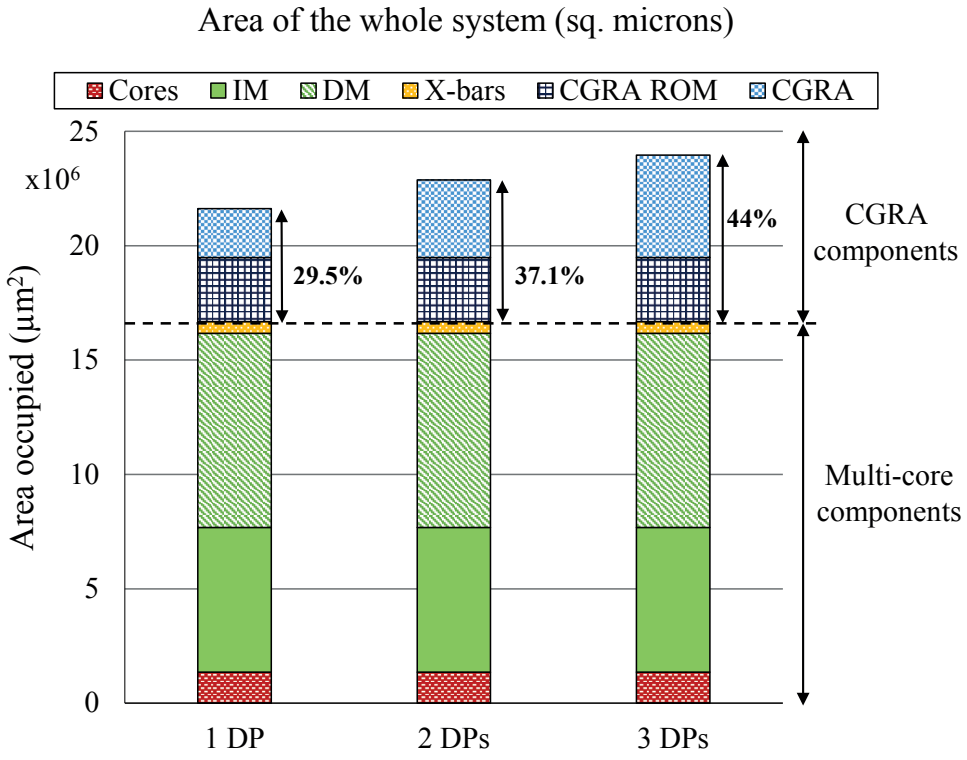


Figure 3.13 – Area breakdown of proposed platform, embedding a CGRA with different SIMD widths.

Figure 3.13 displays the area overhead breakdown of the different system components of the heterogeneous reconfigurable platform. It shows that the CGRA occupies an area of 29.5 %,

37.1 % and 44 % of the total system chip area, for the 1 DP, 2 DP and 3 DP cases, respectively. It can further be derived that the area cost of extending the CGRA to support SIMD kernels is marginal. For example, in the case of the 8L-CS benchmark, the system energy consumption is 29.9 % lower when a CGRA with 2 DPs per RC is used instead of a single-DP CGRA. In that case, the system area penalty incurred due to the additional DP is only 5.8 %.

3.2.3 Interleaved Datapaths (i-DPs) CGRA-based bio-signal processing

Although the SIMD CGRA architecture has been shown to be highly efficient in processing BSP kernels, it is mostly beneficial when the same acceleration is requested by different processors in SIMD mode. However, such a platform may not be highly efficient when kernels are mostly requested by a single core, or by multiple cores in a non-SIMD manner. In addition, BSP kernels usually process loops with high numbers of iterations. This scenario involves a large execution time for most kernels when running on a SIMD CGRA, thereby burning energy that can potentially be saved by optimizing the existing CGRA architecture. One of the promising ways to decrease kernel execution time on the CGRA, while still using an architecture similar to the SIMD CGRA, is to divide the iterations relative to kernels into several parts and to map each part onto a different DP. In this way the effective CGRA execution time can be divided by the number of slices made, with the operations of each DP still governed by the same shared control logic. Orthogonally to the SIMD CGRA, this architecture is thereby optimized to efficiently execute large kernels that are called in a non-SIMD manner in a multi-core platform, or by an architecture which features a single processor.

3.2.3.1 SIMD CGRA limitations

The SIMD CGRA is highly efficient in executing kernels that are called simultaneously by multiple cores, as shown in Section 3.2.2.5. However, in some BSP applications it can happen that the kernels process a large amount of input data and are not called in SIMD fashion by multiple cores. If they are processed with the SIMD CGRA, it is needed to employ a single DP, while power-gating the others. Thus, the entire energy-hungry control logic is dedicated only for a single DP, which is not the aim of SIMD processing. Conversely, if single core BSP architectures are considered, employing SIMD CGRA is an overkill since only one instance of kernels are executed at any given point of time. These observations point toward optimizations in the SIMD CGRA that can effectively parallelize the execution even for kernels called in isolation. To keep taking advantage of the shared logic among multiple DPs, it can be possible to split iterations of a kernel into subparts, each of which can be executed by a DP. As an example, if a loop has 10 iterations, the first 5 can be executed by a DP while the rest 5 are processed by another DP (considering a SIMD width of 2), with both performing the same operations, but on separate data streams. This approach can reduce the execution time of kernels drastically compared to a SIMD CGRA scheduling strategy described in Section 3.2.2.4, potentially saving energy consumption by taking advantage of the shared logic among DPs. The architecture of the resulting Interleaved Datapaths (i-DPs) CGRA is detailed as follows.

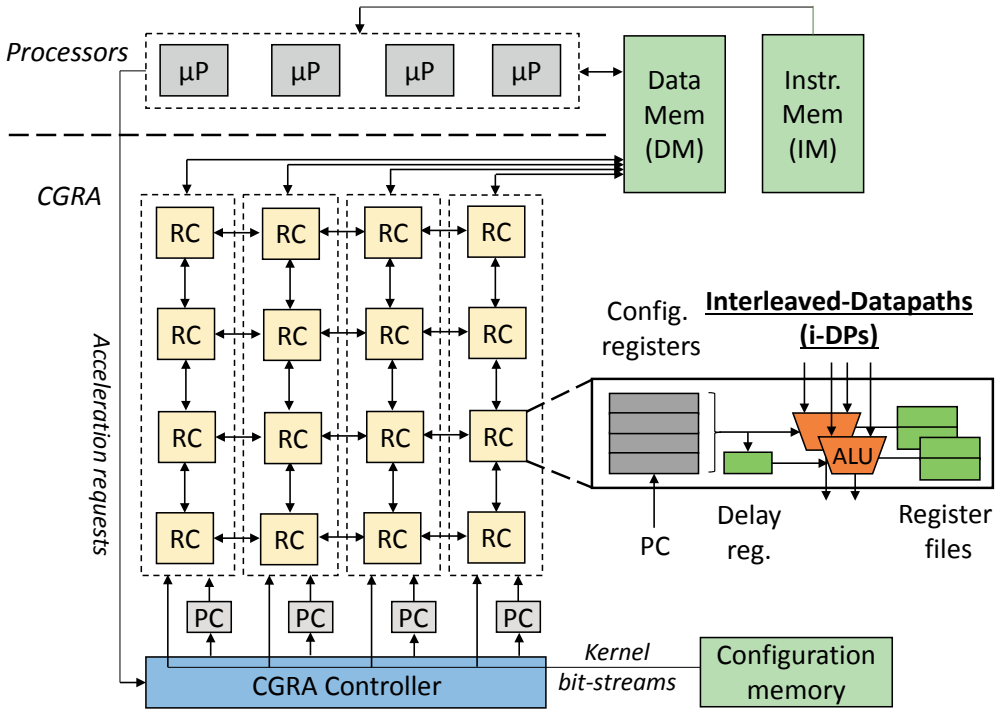


Figure 3.14 – The i-DPs CGRA block scheme, interfaced with a multiprocessor system.

3.2.3.2 i-DPs CGRA structure

The framework of this architecture (shown in Figure 3.14) is based on that of the SIMD CGRA, employing a 4 × 4 RC mesh (cf. Section 3.2.2). The CGRA controller, at the periphery of the mesh, manages acceleration requests from the issuing core(s), mapping them onto available RC columns and programming the corresponding RCs with instruction bit-streams from the configuration RAM. During kernel execution, the operation to be performed at each clock cycle is chosen by column-wise program counters. The active counter value is skewed (employing a delay register) by one clock cycle between the two DPs of an RC. This is done to avoid both DPs from requesting or writing data at the same clock cycle, as such a situation would require multiple port connections between an RC column and the DM, which is not the case in the i-DPs CGRA architecture. In the current platform, transfers between the CGRA and DM are supported by multiplexing the port of the processor that issued the acceleration, hence allowing the transfer, at each clock cycle, of one word of data per active kernel. The rest of the system is similar to the one detailed in Section 3.2.2.3. Small hardware and runtime overheads are required to support interleaved DPs. On the hardware side, delay registers in each RC store one configuration word per DP (32 bits in this implementation). A multiplexer is also required to select, at each clock cycle, the DP that can access the DM. During runtime, scalar constants have to be written for each slice in the local register file of the DPs, and scalar results are transferred back to the DM. In the following sections, it is shown how these overheads are dwarfed by the gains achieved in the execution time of the kernels.

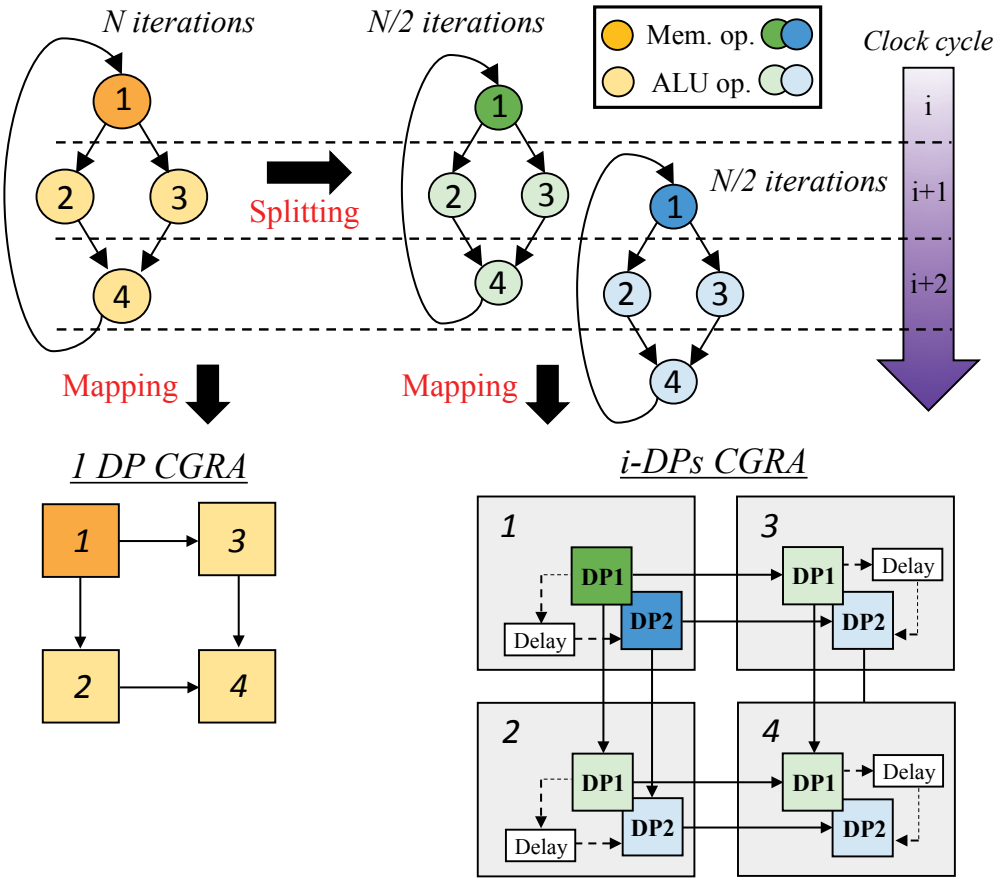


Figure 3.15 – Example of a simple kernel mapped in a single-DP and an i-DPs CGRA. In the latter case, the kernel is split into two slices and mapped on the interleaved DPs of the RCs.

3.2.3.3 Kernel mapping on the i-DPs CGRA

The adopted strategy can effectively map two common types of kernels. First, kernels that do not present loop carry dependencies can perform a slice of all iterations in each DP, without further modifications. Secondly, reduction kernels, which compute one (or few) scalar values from an input array, can be divided into multiple parts and mapped onto different DPs. However, the latter require a wrap-up phase in software after their execution in the CGRA, to aggregate the obtained results. For example, considering a kernel that calculates the maximum and minimum values in a data vector: when split into two parts, each part returns the maximum and minimum values in the related parts of the vector that they process, thereby producing two maximums and two minimums. The software code has to be modified accordingly to compare the two maximums and the two minimums produced by each slice in the final wrap-up phase.

The kernels are selected and mapped in the same manner as explained in Section 3.2.2.4, with the exception that the operations in the second DP are delayed from the first one by one cycle. Figure 3.15 shows the strategy for mapping a simple kernel on an i-DPs CGRA,

considering two DPs. In particular, a given input vector relative to the kernel is split into two parts, each of which is processed simultaneously by the kernel on a different DP. Slicing of the kernel into more parts is also possible, but that involves the usage of further DPs. In the shown example, I have considered a kernel processing a data vector with *N* elements thereby featuring *N* iterations, which has four operations per iteration: a memory load/store in the first clock cycle, two ALU operations in the second cycle, and a single ALU operation in the third one. In a 1 DP CGRA, this kernel would need four RCs to be mapped, with one RC performing the memory operation, while the other three do the subsequent ALU operations (the flow is shown in Figure 3.15, left part). To be mapped into the i-DPs CGRA, this kernel is split into two parts, each with *N/2* iterations. Operations in each slice are then mapped onto the RCs in corresponding DPs (Figure 3.15, right part), following a similar flow pattern as in the 1 DP CGRA. The program counter (thereby operations) of the second DP are delayed by one clock cycle, to ensure that the memory load/store does not happen in the same cycle for both. Finally, the program counter traverses a prologue phase, followed by the iteration and epilogue phases, as described in Section 3.2.2.4.

3.2.3.4 Experimental setup and results

This section details the experimental setup considered in my experiments, the results obtained and the observations made in terms of performance, energy and area consumption.

Framework

A CGRA with a 4×4 RC configuration similar to the one in Section 3.2.2.3 has been considered, with 32 configuration registers per RC. The system components (including processors, memories and the i-DP CGRA) have been characterized at the post-synthesis level, targeting a 65 nm UMC low-leakage technology [105]. The obtained energy parameters were then employed to annotate a cycle-accurate virtual platform (specified in SystemC), allowing fast whole-system simulations of entire applications. Two ECG processing applications have been considered to evaluate the performance of the i-DPs CGRA (cf. Section 2.3.3.3). First, an 8-lead Compressed Sensing (8L-CS) [25] derives a number of random features as linear combinations of input samples [13]. In the targeted implementation, I adopted signal windows of 1024 samples, and a compression ratio of 50 %. The *CS* kernel computes the random indexes using an LFSR. It does not present loop-carried dependencies, and can therefore be mapped on the CGRA in a straightforward manner by distributing its iterations equally among the available DPs. The second application, 6-lead Morphological Filtering (6L-MF), cancels the baseline wander of an ECG record [14]. Its kernels compute the first and second maximum and minimum along the sample windows. They can therefore be divided in slices, each returning the two higher/lower values in a sub-window, with a (short) software wrap-up routine that determines the two final outputs among the values computed by the CGRA. To evaluate the performance and energy of the entire processing system, the i-DPs CGRA was employed in conjunction with a multi-core BSP architecture similar to the one in [21].

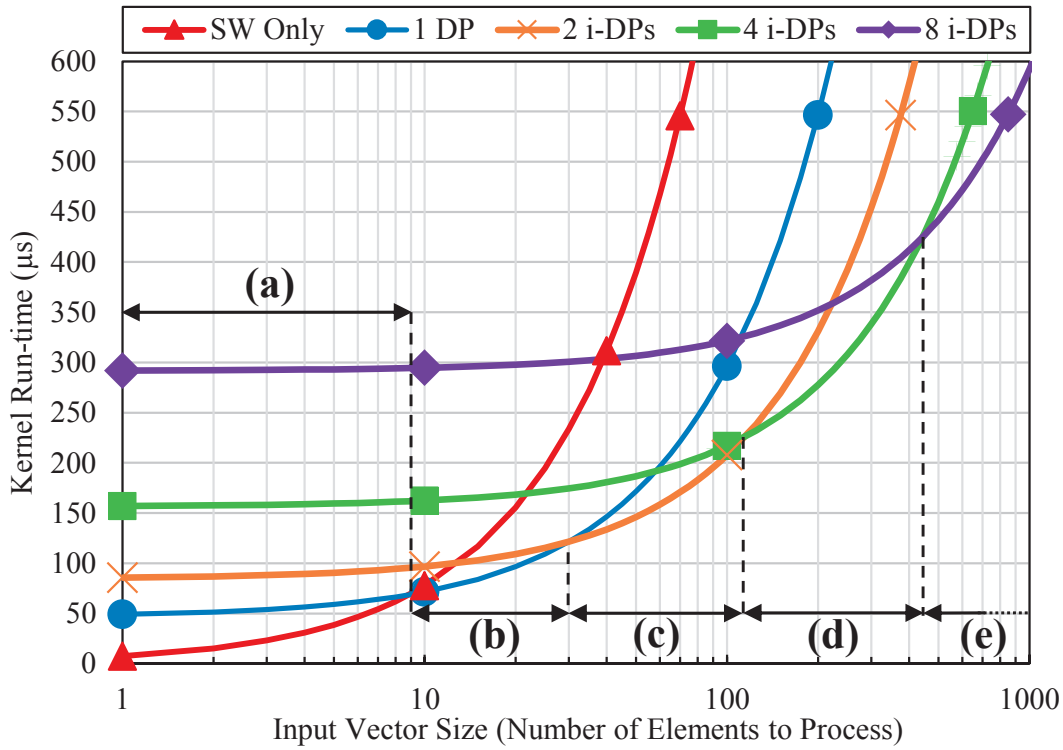


Figure 3.16 – Runtime of *Dbl Min Srch* kernel (first and second minimum element of an array), on a single-DP CGRA and on i-DPs with different widths, varying the input data size.

Performance analysis

The selected kernels can potentially be divided into a large number of slices, to be processed in parallel by the i-DPs CGRA, thereby requiring multiple DPs (corresponding to the number of slices). In such cases, however, the achieved efficiency in terms of execution time may be significantly diminished due to the limited bandwidth between the data memory and the CGRA (which will also require additional delay registers). Moreover, the presence of a large number of DPs may potentially incur a significant timing overhead for the configuration of kernels and the scalar outputs to and from the CGRA for each kernel slice. Furthermore, additional DPs also result in increased energy consumption, and therefore needs to be carefully tailored to avoid canceling out potential energy gains. In addition, for reduction kernels, the higher the number of kernel slices, higher is the software computation time required to select the final values to be considered from the slices.

Figure 3.16 investigates the trade-offs between the size of the input vector to be processed and the number of interleaved DPs, considering the *Dbl Min Srch* kernel (cf. Section 3.2.2.5) as a case study. It shows for each given number of elements in the input vector, the optimal number of DPs that should be employed to ensure the lowest runtime for the kernel, taking into account the software post-computation overhead. From Figure 3.16 it can be observed that for small datasets (region (a)), it is not beneficial to configure and invoke the CGRA, as the

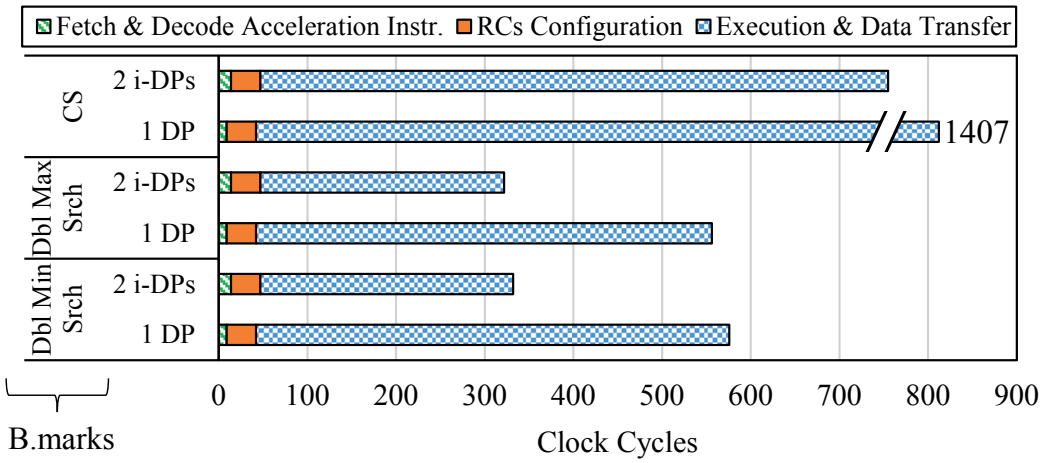


Figure 3.17 – Average kernels runtime (in clock cycles) executing on CGRAs with 1 DP and 2 i-DPs.

related overhead is larger than the benefits deriving from hardware acceleration. Furthermore, as the input data size increases, it becomes more profitable to use a CGRA with an increasing number of DPs (regions (b), (c), and onward). For example, 8 i-DP cells present the lowest runtime if the input size exceeds 446 elements (region (e)). In the considered benchmark BSP applications, the kernel input data have an average size of 100 elements (which is a typical scenario for bio-signal analysis applications). In the rest of this study, I have therefore considered a 2 i-DPs CGRA configuration, as it is the optimal one for this data size according to Figure 3.16. Similar results were also obtained for the other studied kernels.

The execution time of the considered kernels on a single-DP and on an i-DPs CGRA with 2 DPs, without considering the software overhead required by the processors to configure, launch and recover the results from an acceleration, is depicted in Figure 3.17. By adopting i-DPs, a large reduction is obtained in the time required for computing the kernel outputs (*Execution and Data Transfer phase*), which is almost halved. When employing an i-DPs CGRA, there is an increase in the *Fetch and Decode* phase, due to the extra register settings required for the initialization of multiple kernel slices, but its impact is negligible (less than 1.6 % in all cases). These gains at the kernel level are in turn reflected at the system level as both the considered applications spend a considerable amount of their execution time in processing the kernels (28 % and 88 %, for 6L-MF and 8L-CS, respectively).

Energy analysis

At the level of individual kernels, Figure 3.18 shows that the i-DPs CGRA is able to reduce the energy budget significantly over both a SW-only and a single-DP CGRA equivalent execution. In particular, the energy overhead for executing the *CS* kernel is reduced by 34 %, compared to a 1 DP CGRA. While for the two kernels (*Dbl Min Srch* and *Dbl Max Srch*) present in the 6L-MF

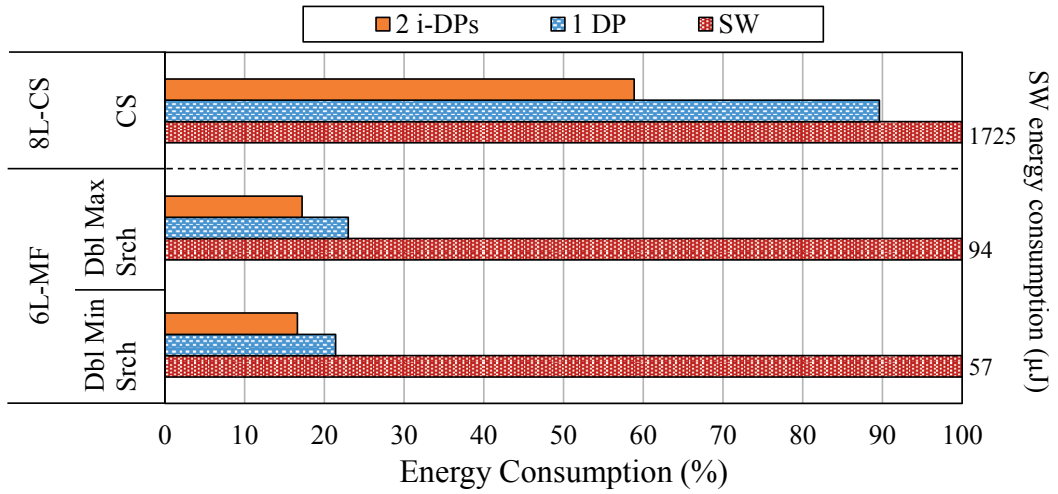


Figure 3.18 – Energy consumption of the different kernels. For each kernel, the bars are normalized to the SW-only energy.

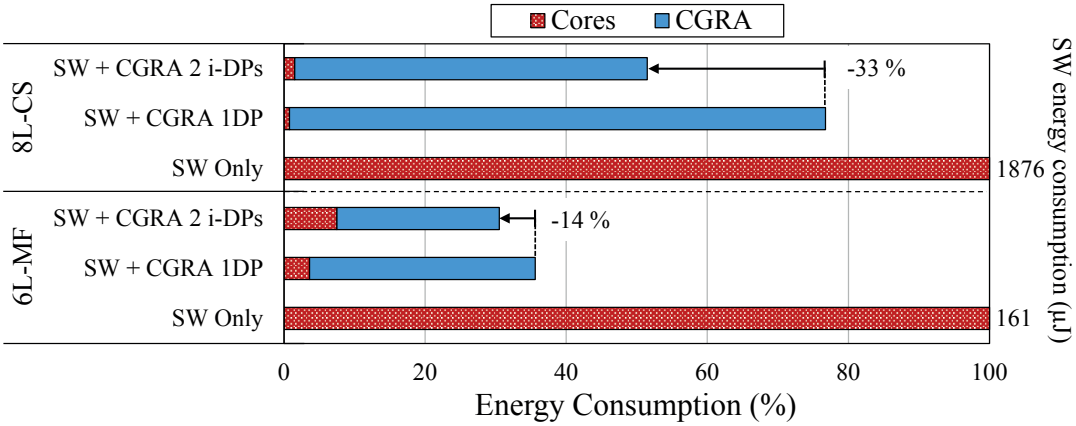


Figure 3.19 – System energy consumption of the different benchmark applications, normalized to the SW-only consumption for the part of the application transfered to the CGRA.

application, the savings are 22 % and 25 %, respectively. The energy benefits in this case are derived from two main sources: (i) the increased idle times are exploited to aggressively clock-gate idle system components, resulting in a decrease in the dynamic energy consumption; (ii) the i-DPs CGRA configuration results in a high ratio between the CGRA logic devoted to computing (RCs) and that used to control the execution flow (configuration registers).

Figure 3.19 compares the energy consumption of the entire system for the part of the application that is transferred to the CGRA. The baselines for comparing the efficiency of the i-DPs CGRA is a multi-processor-only architecture [21] and a WBSN platform featuring a single-DP CGRA [150] along with the cores. For both cases, the i-DPs CGRA results in significant en-

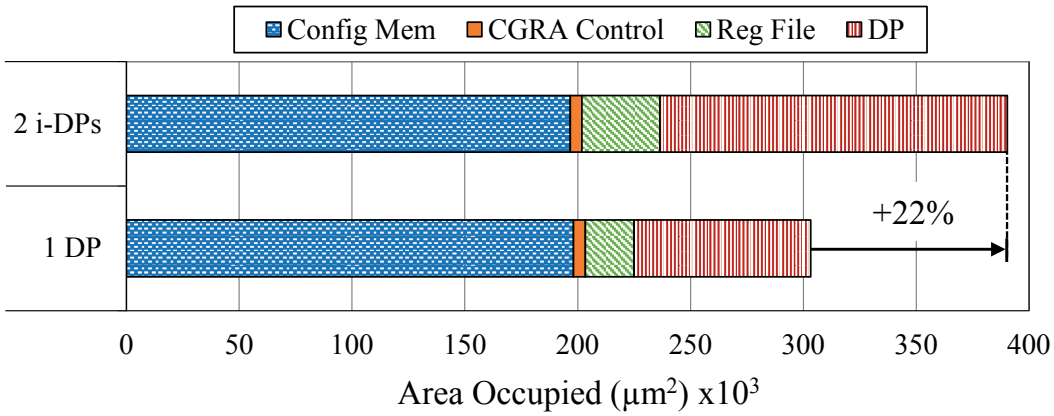


Figure 3.20 – Area breakdown of the i-DPs CGRA compared to a traditional 1 DP CGRA.

ergy optimization for both the considered applications, even when the software overhead for launching the kernel requests and post-computation are taken into account. Again, this happens mainly due to the significantly faster processing of the kernels in the i-DPs CGRA than in the cores and also since the cores are clock-gated when acceleration issued by them are accepted by the CGRA.

Figure 3.19 also points out the energy benefits over a single-DP CGRA, when the i-DPs CGRA is used. Although the software overheads for the i-DPs are almost doubled compared to a 1 DP CGRA, its usage still results in significant reduction of the total system energy overhead with respect to the 1 DP CGRA case. Notably, 14 % and 33 % reductions in the system energy budget are obtained for 6L-MF and 8L-CS applications, respectively. The resulting energy envelopes in the i-DPs CGRA case are nevertheless much smaller compared to a processor-only alternative (*SW Only* in Figure 3.19).

Area analysis

Figure 3.20 analyzes the silicon real estate required by the various CGRA components in the case of i-DPs CGRA compared to a traditional 1 DP CGRA. The overhead of the configuration memories of RCs and the CGRA control logic is consistent in both architectures since these are shared between the multiple instances of DPs. The net area increase entailed by adding a DP to an RC amounts to 22 % over a 1 DP CGRA. The area occupied by register files is also increased in the case of i-DPs CGRA since the number of DPs is doubled (thereby featuring double the number of register files), compared to a single-DP CGRA.

Finally, since the configuration memory is the dominant area consumer, whose value is independent of the number of DPs, doubling the number of DPs has a limited impact on the total system area occupation. This observation supports the employment of an i-DPs CGRA over a single-DP equivalent as large energy reductions are possible with the addition of one DP.

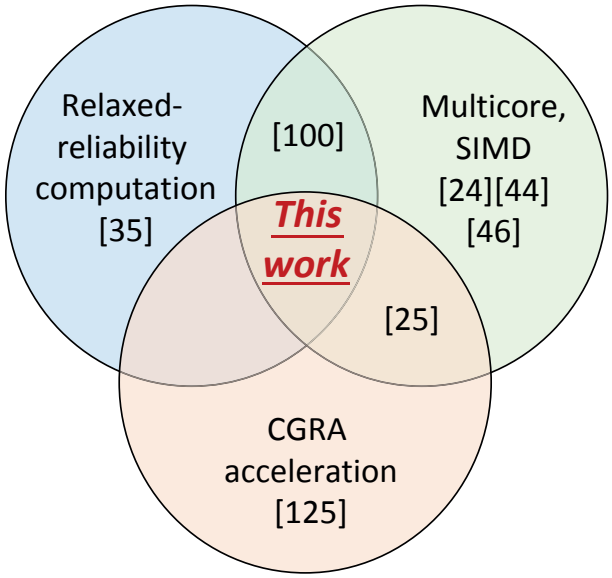


Figure 3.21 – Previous works have explored, in isolation, multi-core processing, CGRA acceleration and inexact computing in the field of ultra-low power bio-signal processing. In this section, I explore their ensuing synergies.

3.2.4 Reconfigurable WBSNs with relaxed-reliability computation

While reconfigurable architecture-based optimizations substantially increase the energy efficiency of WBSNs, further improvement may be achieved by jointly leveraging CGRA-based acceleration and relaxed-reliability computation (cf. Chapter 2). In this section, I study the application of relaxed-reliability paradigm on the CGRA-based heterogeneous WBSN system described in Section 3.2.2.3, derived from inexact computation arising from ultra-low voltage scaling.

3.2.4.1 Heterogeneous and relaxed-reliability computation

In Section 3.2.2, I have shown that the execution of computational hot-spots (kernels) in BSP algorithms can be effectively performed by CGRA accelerators. Thanks to the faster and more efficient execution achieved with the aforementioned technique, WBSN systems can prolong the sleep mode of processors, consequently reducing their energy consumption.

An orthogonal strategy directed at increasing the energy efficiency of WBSNs is inexact computation, that waives the requirement of exact results by introducing relaxed-reliability computing paradigm. By trading off accuracy of the produced results, this strategy aims at increasing the energy efficiency, and is acceptable in application domains where low energy consumption, processing deadlines and throughputs are more important than the preciseness of the produced output. BSP applications are tolerant to some amount of errors, as they are inherently corrupted by noise from different body parts, and as their outputs often are qualitative

or statistical in nature. These applications are therefore able to tolerate a non-zero probability of runtime failures, as long as the produced output retains appropriate clinical significance and the system is never trapped in unrecoverable or erroneous states.

In this context, voltage over-scaling is an efficient choice to reduce the energy consumption of WBSNs, while introducing inexactness in them as system components become error-prone when supplies are lowered. As discussed in Section 2.1.1, the combinational elements of WBSNs, such as Arithmetic Logic Units (ALUs) and registers (flip-flops) are less sensitive to voltage scaling, particularly at low frequencies (few MHz) which are typical in the BSP domain. However, the robustness of SRAMs, which are typically used for WBSN memories, is hampered while operating at Near-Threshold Voltages (NTV), especially when runtime supply voltage fluctuations (*droops*) are taken into account [153]. My focus in this study is therefore on memories (data and instruction) as sources of failures, manifested as bit-flips in WBSN systems.

As a first step, to study the effects of voltage over-scaling when approaching ultra-low levels, I have analyzed the application-level effects of the resulting bit-flips in the memories. Then, I have employed architectural techniques such as memory protection, input and intermediate buffer surveillance, and monitoring of runtime synchronization between computing elements, to minimize the impact of failures and to guarantee system recovery (cf. Section 2.3.3.2).

In this research, I have explored the joint application of CGRA-based acceleration and relaxed-reliability computation on WBSN architectures in order to leverage the benefits derived out of both strategies. Previous works in the BSP domain have applied the approaches of multi-core SIMD processing [24] [154] [44], CGRA-based acceleration [125] and relaxed-reliability computation [100] independently (except in the case of WBSNs with multi-core and CGRA processing [25]). Herein, I combine all these strategies, as depicted in Figure 3.21, to achieve additive gains in the resulting energy efficiency, while aiming to keep the ensuing error in the output within permissible limits.

3.2.4.2 Experimental setup and results

To evaluate the joint application of the energy-saving strategies, a WBSN platform similar to the one described in Section 3.2.2 has been employed (Figure 3.22). The target architecture features a multi-core processing system along with a domain-specific CGRA, similar to the one presented therein. On the processor side, there are eight TamaRISC cores (details in Section 3.1.1.1), which are interconnected through combinational crossbars to separate data and instruction memory banks. Each processor implements a Harvard-like architecture [155], supported by a three-stage pipeline, which can be clock-gated by a *synchronizer* unit while waiting for another processor to finish its task or when a kernel acceleration called by it is running on the CGRA. At the application level, the processors rely on several synchronization instructions to support SIMD execution modes and producer-consumer relationships between cores [21]. An additional special instruction enables the request for a kernel execution in the

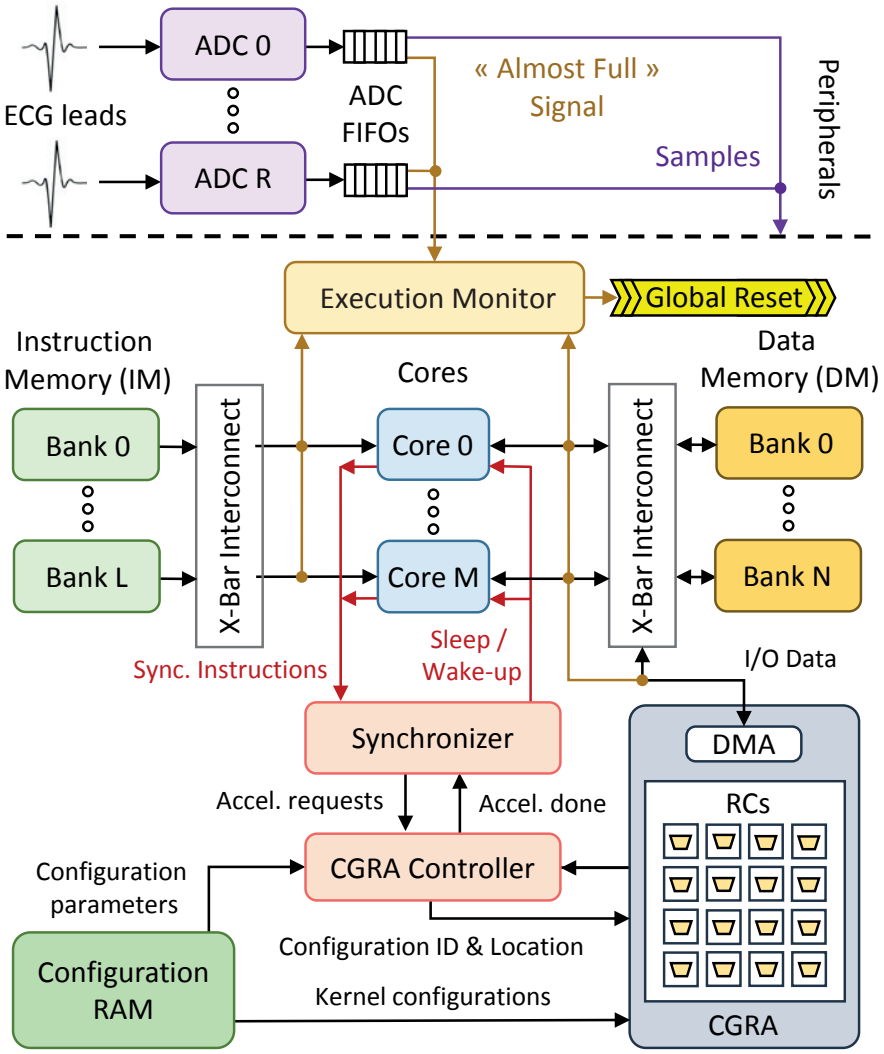


Figure 3.22 – High-level block scheme of the proposed heterogeneous architecture with relaxed-reliability requirements.

CGRA [25]. A *CGRA Controller* unit handles acceleration requests from processors, enabling both concurrent and sequential access to the CGRA. Requests are stored in a dedicated queue and are mapped on the CGRA mesh as soon as enough resources are available. An *Execution Monitor* module resets the platform if an incorrect system state (as a result of a bit-flip in memory) is detected.

Similar to Section 3.2.2.1, the CGRA mesh consists of Reconfigurable Cells (RC) arranged in a 4×4 configuration with near-neighbor connections with torus wrap-around. Each RC consists of a Configuration Register (CR) file featuring 16 words, each of 32 bits, which is in charge of storing the instructions to be executed by an RC when processing a kernel. RCs support SIMD execution, by featuring two DataPaths (DP) governed by the same control logic; with the 2 DP configuration chosen as it results in the best processing efficiency (cf.

Section 3.2.2.5). Thus, the same kernel requests originating from different processors (and hence operating on separate input data) can be efficiently processed in the SIMD mode [25]. When not operating in the SIMD mode, kernels can be concurrently mapped on separate CGRA columns. A framework similar to the one detailed in Section 3.2.2.3 is also employed in this study, with the CGRA undergoing a configuration phase followed by an execution phase (for details see Section 3.2.2.4).

I introduce an *Execution Monitor* (EM) module that is in charge of detecting memory failures and reseting the system to resume operation from a safe state. Traditionally, large (and costly in terms of energy) guard-bands are employed for the voltage supply, as even small fluctuations in it can result in memory failures at NTV [94]. The EM performs hardware- and software-level checks to detect errors, thereby eliminating the need for guard-bands. As in Section 2.3.3.2, this unit implements a number of error monitoring and recovery strategies, described in the following.

- ***Runtime coherence:*** Detects inconsistencies in the synchronization sequence and in the execution flow of the cores. It ensures that synchronization events are processed in a last-in first-out manner for each core. A violation of this pattern indicates an erroneous execution flow (e.g., due to a jump to an incorrect IM location).
- ***FIFO surveillance:*** Monitors overflows and underflows in ADC and inter-processor FIFOs, which indicate that a processor is stuck (e.g., due to a bit-flip causing the execution of an infinite loop).
- ***Data Memory (DM) protection:*** Detects illegal data writes to read-only DM locations.
- ***Instruction Memory (IM) footprint violation:*** Detects erroneous jumps to non-initialized IM addresses.

Upon detection of an error, the EM activates a global reset signal to restart the platform from a safe initial state. These strategies can be realized with a small hardware overhead ($\approx 1\,\%$ of the total area and $\approx 6\,\%$ of the energy for the considered system), and without any performance penalties.

Application-level errors

Bit-flips in the memory subsystem ensuing as a result of voltage over-scaling in combination with runtime droops, manifest as application-visible errors that have been classified as follows (for details see Section 2.3.2.2) [100]:

- ***Masked errors:*** A bit-flip with no visible effects on the execution flow of the application or on its outputs.

- ***Silent Data Corruption (SDC) errors:*** A bit-flip which has no visible effects on the execution flow of the application, but degrades the output results.

- ***Unrecoverable errors:*** A bit-flip leading to a critical runtime change in the execution flow of the application (e.g., memory footprint violation, execution stuck in an infinite loop), triggering the assertion of a system reset from the EM to recover a consistent state.

Experimental results

This section first details the experimental setup that has been adopted to evaluate the proposed inexact heterogeneous BSP architecture. Then the obtained results are discussed, focusing on the trade-offs between the achieved energy efficiency and the output quality degradation. While, herein, I employ a SIMD CGRA for the application of relaxed-reliability computation, the same technique can be extended in the future to the Interleaved-DPs (i-DPs) CGRA architecture (cf. Section 3.2.3).

Framework

To evaluate the inexact and heterogeneous WBSN architecture, I have employed a hybrid framework combining a post-synthesis model and a cycle-accurate simulator of the system, similar to the one in Section 3.2.2.3, represented in Figure 3.22. The RTL netlist, cycle-accurate (SystemC) model and compiler of the processors are derived using Synopsys ASIP Designer [104]. The complete system includes 8 processors, a DM of 64 kB (32 K words of 16-bit each) and an IM of 96 kB (32 K of 24-bit words). The CGRA configuration RAM has a size of 6 kB (1.5 K of 32-bit words), which is sufficient to store the instruction bit-streams of the considered kernels.

The CGRA, the EM and the memory subsystem have been developed as HDL modules (using pre-synthesized models from the memory vendor), and the whole system has been synthesized using Synopsys Design Compiler [149] with a 65 nm UMC low-leakage library [105]. The execution of kernels in the CGRA were simulated with Mentor Graphics ModelSim [148]. The corresponding switching activities were then used to accurately derive the power consumption of the CGRA, considering complete power-gating of unused columns at runtime. Similarly, the performance and power consumption of the various components of the inexact system were derived by running small synthetic benchmarks (both with and without SIMD support) on a post-synthesized design. The obtained energy profiles, along with the kernel runtimes on the CGRA, were then used in the SystemC simulator of the system, thus enabling faster simulations.

To validate the experimental setup, two commonly used BSP application benchmarks were employed (cf. Section 2.3.3.3) [25]:

- ***Eight-lead Compressed Sensing (CS)***: This is a highly parallel application to perform lossy encoding of eight ECG leads, where each lead is processed by a different core.
- ***Four-lead Morphological Filtering, combined with CS (MF-CS)***: The MF stage processes four ECG leads in parallel (one core per lead), removing high- and low-frequency noise with morphological operators. The filtered ECGs are then compressed by the remaining four cores, producing a filtered and compressed ECG as output.

Both benchmarks receive as input ECG signal windows of 1024 samples acquired at 500 Hz and extracted from the T-Wave Alternans Challenge database [156]. After the 50% CS compression, signal windows of 512 samples are delivered as output. I have employed VARIUS-NTV [96] to obtain SRAM failure rates due to stability and timing violations. Furthermore, voltage droops have been modeled as random fluctuations, following a normal distribution centered on the nominal supply value V_{dd} and with variance σ_n, where a higher value of σ_n indicates a larger voltage fluctuation (details in Section 2.3.2.1). Three types of memory access corruptions are considered: data read, data write and instruction read. For each benchmark and for each type of memory access failure, a representative set of 1000 input windows was processed, with each simulation incurring in a bit-flip at runtime. Then, I observed the impact of the bit-flip at the application level (i.e., if it caused an SDC error, an unrecoverable error, or if its effect was masked).

Energy/quality-of-service trade-offs

In order to asses the system performance from a relaxed-reliability computing perspective, the ratio of the windows that terminate the required processing to the total number of ECG windows processed, is chosen as a metric for Quality-of-Service (QoS) of the system. In my experiments, I have pessimistically assumed that an entire ECG window is discarded upon the detection of any error by the EM, and the system execution is restarted from a new window. The experiments have been performed at varying voltage supplies between the levels corresponding to 0 % and 100 % QoS, with different voltage droop variances. Figure 3.23 details the resulting QoS of the system depending on the supply voltage in presence of voltage droops. It can be observed that the QoS remains high (> 90 %) till the supply voltage is reduced to 0.89 V and 0.93 V (at σ_n = 70 mV), for CS and MF-CS, respectively. At these supply ranges, the system withstands bit-flip errors in the order of thousands per hour (11 000 and 3000, respectively). In comparison, an exact system that guarantees less than one bit-flip per year needs a supply voltage of 1.3 V. This implies reduction in the supply voltage by 31.5 % and 28.5 % for CS and MF-CS applications, respectively. It can be noted that compared to Section 2.3.3.3 supply voltages requirements guaranteeing a high QoS are lower, as in the current case the number of SDC and unrecoverable errors are diminished in comparison. This is because when the CGRA is used along with processors, the number of accesses to the IM is lowered, as cores outsource the execution of kernels to the CGRA, thereby reducing errors in the application output.

Figure 3.24 shows the energy consumed per valid window of computed ECG (i.e., the ones

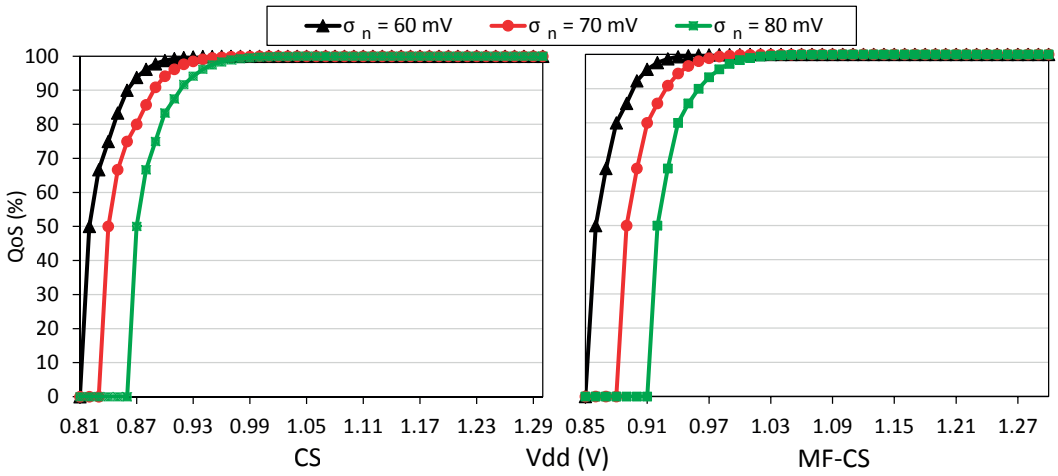


Figure 3.23 – Quality-of-Service (QoS) for different voltage supply values and droop variations for the proposed architecture.

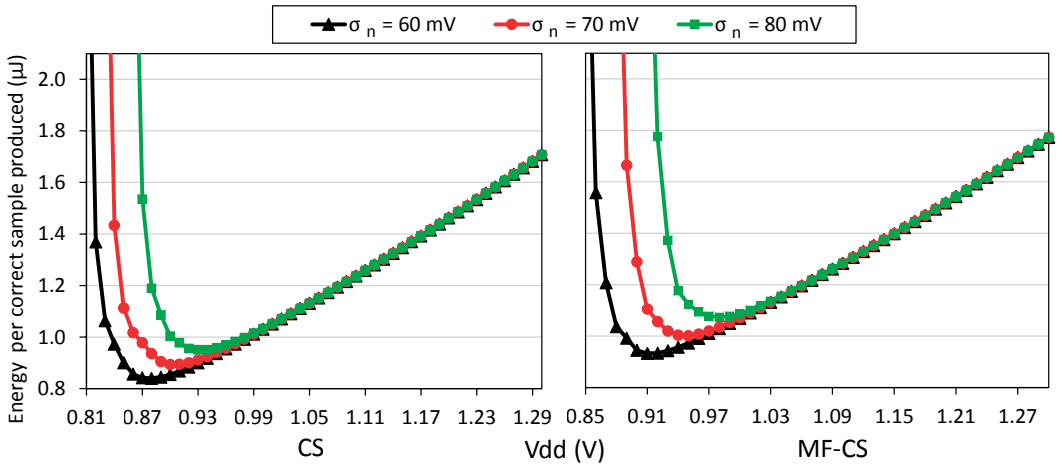


Figure 3.24 – Evolution of the energy consumption per compressed sample produced, for different voltage supply values and droop variations.

that are not discarded due to a reset assertion by the EM). Lower voltages present a high energy consumption since a large number of windows are discarded due to a high volume of errors, thereby leading to wasted energy. On the other hand, at high voltage ranges, the energy consumed is also greatly increased, even though the number of valid windows is higher. Thus, the optimal supply voltage can be inferred as the point in which the amount of energy per valid window is minimized. Considering an average σ_n = 70 mV, the optimal V_{dd} values are 0.9 V and 0.95 V, which guarantee a very high QoS of 94.1 % and 96.7 % for CS and MF-CS, withstanding 6500 and 1050 bit-flips per hour, respectively.

To assess the effect of undetected SDC errors on the received signal quality, the Signal-to-Noise Ratio (SNR) and the Percentage Root-mean-square Difference (PRD) of the received signals were calculated [13]. They were then compared to error-free executions, according to the

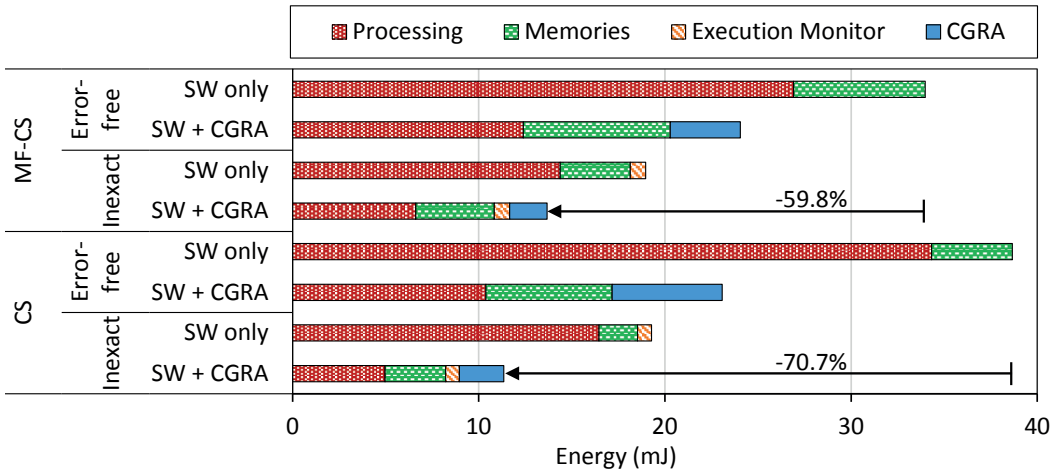


Figure 3.25 – Detailed energy consumption breakdown for the studied system.

expected amount of correct windows and windows affected by SDCs (computed by the system depending on the voltage supply level). Experimental results show that, when working at the optimal V_{dd}, the system's SNR remains very high (65.8 dB for CS and 79.3 dB for MF-CS, respectively), with a net PRD falling in the range of very good signal quality [157] (0.36 for CS and 0.11 for MF-CS).

Finally, Figure 3.25 compares the total system energy consumption of the different platforms, for the two considered applications. The reference baseline is an exact WBSN system (running at 1.3 V) without any CGRA acceleration. The first comparison is made with a system working at the same V_{dd} but with CGRA support. By transferring the execution of kernels to the accelerator, the time to process a sample of signal is reduced by 36.3 % and 41.7 % on average for CS and MF-CS, respectively. This saves a significant part of the processing energy as the cores are in sleep mode and the CGRA executes the kernels with high energy efficiency, leading to savings of 40.3 % and 29.2 % for CS and MF-CS, respectively.

On top of CGRA acceleration, by applying relaxed-reliability computation, further energy savings can be obtained due to voltage over-scaling. When running at the energy-optimal V_{dd} obtained from Figure 3.24 (0.9 V and 0.95 V for CS and MF-CS, respectively), the inexact system without CGRA increases its energy efficiency by 50.2 % and 44.2 %, for the two benchmarks. Finally, by combining a CGRA-based WBSN platform with relaxed-reliability computation, 70.7 % and 59.8 % energy consumption savings are obtained for CS and MF-CS, respectively, over an exact system without CGRA acceleration.

These observations therefore showcase that the energy savings obtained by employing relaxed-reliability computation in tandem with CGRA-based acceleration for WBSNs are much higher than the overheads induced by the additional components that are required to support these two strategies (the *Execution Monitor* for relaxed-reliability computation and the *CGRA* for heterogeneous WBSNs).

3.3 Resistive RAM-based heterogeneous WBSN architectures

3.3.1 Introduction

Architectural and algorithmic optimizations for low-power WBSN operation greatly contribute to the lowering of energy costs. However, their effects are limited by the loss of reliability of traditional SRAM memories at ultra-low voltages, and due to the increase in leakage power at smaller transistor technologies. Moreover, in cases of BSP applications such as for ECG, WBSNs spend a significantly high amount of time in the *idle* mode, thereby making leakage power consumption extremely critical. As an example, a standard ECG compression and filtering application [13] can have more than 90 % of the total execution time spent in the idle mode, where no computation is done. In this period, the system components end up consuming significant amount of leakage power, representing up to 88 % of the overall power consumption [30]. As WBSNs traditionally use volatile SRAM-based memories, turning off the system during idle states to save on leakage is not an option. On the other hand, traditional Non-Volatile Memories (NVM) are not used in WBSNs as they are typically slower and present a much higher energy budget, which is not desirable [37]. As an example, embedded flash (eFlash) memories have a high process cost, high energy consumption, limited reliability while being not available yet in advanced nodes [158] [159] [160].

Emerging non-volatile Resistive RAMs (ReRAM) present a good solution to this issue, as they consume a low amount of power, while being much faster than flash memories and integrable with traditional WBSNs [30]. They offer a low-cost fabrication process (back-end-of-line integration with advanced CMOS [161]) and lower power consumption compared to eFlash. However, ReRAMs suffer from limited endurance capabilities, which occurs as a result of multiple writes to their bits. While their endurance is better than eFlash (10^4- 10^6 writes) [162], it is still limited to a relatively low number of writes (10^6- 10^8 writes) [163]. This calls for energy-efficient solutions like block replacement or wear-leveling to address this shortcoming in order to prolong their lifetime, and therefore the operation periods of WBSNs.

In this study, I show that a heterogeneous memory hierarchy including ReRAMs instead of SRAMs can achieve substantial power savings for a real-world Bio-Signal Processing (BSP) application. Furthermore, I conduct a study of ReRAM endurance, as a function of the number of writes to it, exploring the additional ReRAM capacity required to ensure device functionality for up to one year, considering a lightweight block replacement mechanism.

3.3.2 Emerging ReRAM memories

Table 3.3 compares SRAMs and ReRAMs, considering various memory parameters [30] [164]. As it can be observed, ReRAMs consume substantially smaller area compared to SRAMs. Furthermore, their non-volatile characteristic drastically reduces the leakage power at the system level. Most importantly, this characteristic also enables aggressive runtime energy-saving strategies which can be leveraged in application platforms (such as BSP) where most

Table 3.3 – Comparison of various memory parameters for SRAM and ReRAM.

	SRAM	ReRAM
Read Energy (pJ/bit)	Low (< 1)	Low (< 1)
Write Energy (pJ/bit)	Low (< 1)	Med (5-20)
Leakage	High	Very low
Endurance (writes/bit)	High (> 10^{15})	Low (10^4-10^8)
Cell Size (μ^2 at 28 nm)	0.12	0.03
Non-volatile	No	Yes
Read Latency (ns)	Low (< 10)	Low (< 10)
Write Latency (ns)	Low (< 10)	Med(10's)

of the time is spent by the processing cores in the idle state. Indeed, thanks to their low-cost fabrication process (back-end-of-line integration with common materials [161]), filamentary ReRAMs such as Oxide-based ReRAM (OxRAM) technologies are already included in low-power micro-controllers replacing embedded flash technologies [163].

ReRAM technologies rely on the controlled creation and destruction of a conductive filament inside an insulating material layer. Write operations are performed by applying a programming voltage (V_{prog}) across the bit-cell and controlling the current going through the ReRAM (programming current, I_{prog}) [165] [161] while waiting for the resistance state to switch to a Low or High Resistance State (LRS, HRS), i.e., t_{prog}. As widely shown in the literature, an exponential relationship exists between t_{prog} and V_{prog} [161] [166], while the LRS value is mainly controlled by I_{prog}. On the other hand, read operations are performed by sensing the resistance value (HRS or LRS). In terms of area consumption, 1-Transistor 1-ReRAM (1T1R) bit-cells using thin oxide transistors and standard logic layout rules have been demonstrated down to $12F^2$ ($0.0308\mu^2$) [164]. Compared to high density SRAM bitcells, ($0.12\mu^2$) [167], they enable a 4x area reduction with the subsequent die cost contraction.

However, from Table 3.3, it is evident that ReRAM endurance is orders-of-magnitude lower than SRAMs, starting to show permanent bit-flips after a relatively low number of writes to them ($\approx 10^6$) [141]. Figure 3.26 demonstrates the reliability of a commercial 64 kB ReRAM module [163], and it can be observed that after $\approx 1.8 \times 10^6$ writes per word, all the words in the chip are damaged. This situation advocates the use of heterogeneous memory hierarchies, where first-level page buffers reduce the number of writes to second-level ReRAMs.

In this study, I adopt a heterogeneous WBSN memory architecture employing ReRAMs, performing a technologically accurate power characterization of these emerging memories. Furthermore, I perform the execution of a commonly used BSP application on such a platform to study the endurance-related effects on ReRAMs due to write operations in their cells. Finally, I derive the additional memory that is required to ensure extended device usage, when a block replacement strategy is employed to counter errors occurring as a result of ReRAM endurance.

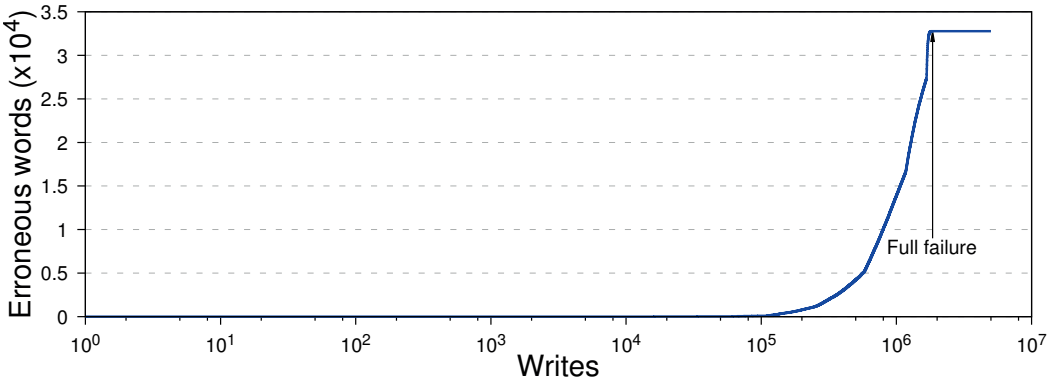


Figure 3.26 – Reliability of a commercial 64 KiB ReRAM module. After $\approx 1.8 \times 10^6$ writes per word, all the words in the chip are damaged. A word is considered damaged when at least one bit in it is flipped and stuck at the changed value.

The experimental setup adopted to validate the proposed strategy is described in the following.

3.3.3 Experimental setup

This section first details the methodology adopted to characterize the ReRAM programming energy, followed by the description of the target heterogeneous platform and the system power characterization. Then, it describes the adopted experimental framework, explaining the obtained results.

ReRAM programming energy characterization

To derive the power consumed while writing to or reading from a ReRAM cell, it is important to characterize its programming energy, which is dependent on the period and amplitude of the programming pulse. In this work, I have considered a 50 ns cycle time for the full system (i.e., 20 MHz), which is in accordance with the literature. As shown in [161], a few ns of programming time (t_{prog}) can be achieved for both *set* (HRS to LRS) and *reset* (LRS to HRS) operations while considering 1 V to 1.5 V programming voltage (V_{prog}). In this context, I have assumed that 100 % of the bit-cells can be programmed in one clock cycle. To reduce distribution tails and to satisfy this assumption, specific programming strategies such as adaptive programming voltage [168] can also be considered. Finally, I have considered I_{prog} = 100 µA to achieve a sufficient HRS/LRS ratio, a low variability in the LRS state and a retention of several years [161].

As shown in Figure 3.27, the programming operation has been modeled in two distinct phases separated by a sharp switching: *(i)* the pre-programming phase, in which the ReRAM is in its previous resistive state; *(ii)* the post-programming phase, in which the ReRAM is in the aimed resistive state. As a margin, I have considered that no state switching happens during the first and the last 5 ns. Summing up the programming energies corresponding to phases

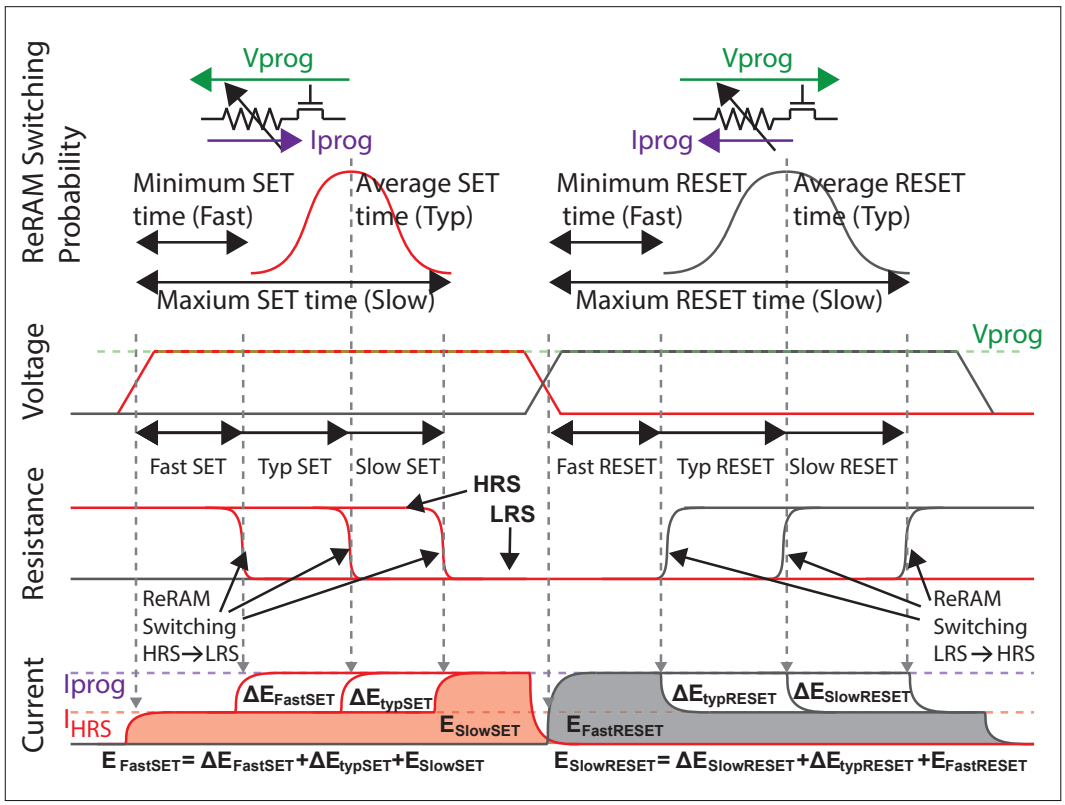


Figure 3.27 – Programming operations in ReRAM technologies (*set* in red and *reset* in gray) with detailed voltage current and energy considerations. ReRAM corners (fast, slow) are also defined.

(i) and (ii) determines the energy consumed during the full programming pulse. From this model, four corner cases were identified: fast *set*, slow *set*, fast *reset* and slow *reset*. Assuming balanced programming operations (i.e., $I_{set} = I_{reset}$), they can be reduced to two corners: the Best Case (BC) corresponding to a slow *set* (thereby consuming low amount of current) and fast *reset*, and the Worst Case (WC) corresponding to fast *set* and slow *reset* (representing a high amount of current). Considering a $t_{prog} = 50\,\text{ns}$, $V_{prog} = 1.2\,\text{V}$ and $I_{prog} = 100\,\mu\text{A}$, the energy consumed in each corner is determined using Equations 3.1 and 3.2 as $E_{BC} = 1.248\,\text{pJ/bit}$ and $E_{WC} = 5.472\,\text{pJ/bit}$.

$$E_{set} = V_{prog} * (t_{prog} * I_{HRS} + (t_{cycle} - t_{prog}) * I_{prog}) \tag{3.1}$$

$$E_{Reset} = V_{prog} * (t_{prog} * I_{prog} + (t_{cycle} - t_{prog}) * I_{HRS}) \tag{3.2}$$

At the end of the programming operation, if any bit in the word did not successfully switch, the write amplifier notifies the memory controller and the word is marked as permanently defective.

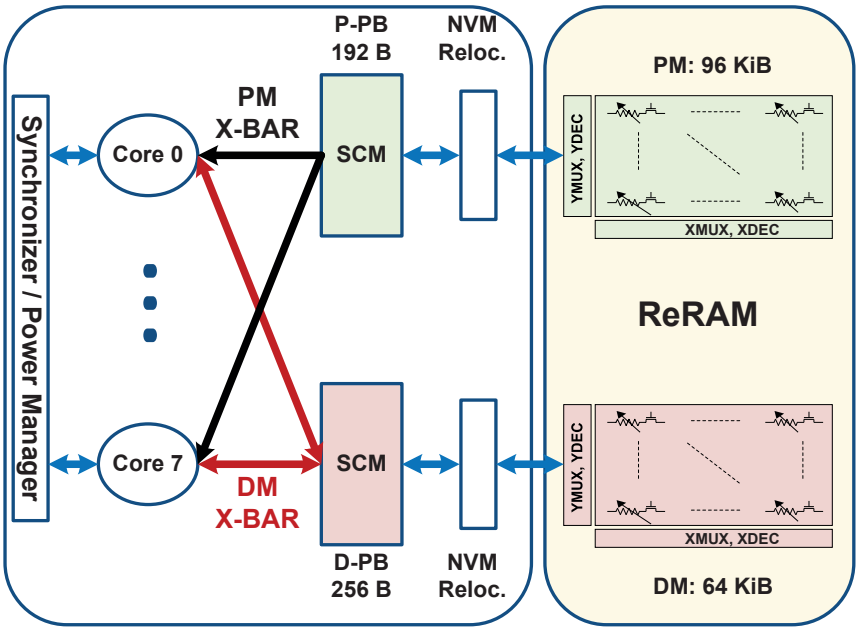


Figure 3.28 – Schematic representation of the heterogeneous architecture.

Target heterogeneous platform

To study the effects of ReRAM endurance, I have used the heterogeneous BSP architecture shown in Figure 3.28, which consists of eight TamaRISC cores featuring a three-stage Harvard-like architecture [101], that have been generated with Synopsys ASIP designer [104]. I have also considered a memory architecture inspired by the work in [30], featuring a hybrid system partitioned into a lower tier with page buffers and a higher tier with the ReRAM main memory. In such an arrangement, the main memory, along with its access transistors, is integrated on top of the processing cores and the related circuitry by employing TSVs. A configuration with eight 8-word Program Page Buffers (P-PB, 24 bits per word) and sixteen 8-word Data Page Buffers (D-PB, 16 bits per word) was employed.

The (non-volatile) main memory is divided into 96 kB of Instruction Memory (IM) and 64 kB of Data Memory (DM). A Memory Management Unit (MMU) is in charge of exchanging the pages between the (volatile) page buffers and the main memory, adopting a least-recently-used strategy. The MMU can be turned off to bypass the page buffers, thereby considering a system with main memory only. Furthermore, each core is interfaced through logarithmic crossbars with the page buffers, which collectively act as a cache to the main memory. Finally, a *synchronizer* module monitors the activities of the cores and coordinates their execution, thereby maximizing the lock-step execution mode. This module is also in charge of power-gating the entire system when the cores are in an *idle*-state (deep sleep mode). The page buffers are never powered off, to decrease the number of writes to the ReRAM required to back them up when power-gated, thereby to increase their endurance.

Table 3.4 – Energy consumption per access and leakage power for each type of memory used in the different platform configurations.

	SRAM		ReRAM		PB	
Size	64 kB	96 kB	64 kB	96 kB	256 B	192 B
Read (pJ/bit)	0.229	0.287	0.264	0.266	0.028	0.024
Write (pJ/bit)	0.194	X	5.472	X	0.032	0.028
Leakage (μW)	1.476	2.160	0.033	0.053	0.004	0.003

In order to study and compare the power efficiency of the hybrid system, four configurations were considered, all of which are assumed to work at 1 V supply and 20 MHz clock, whose time period (50 ns) is higher than the read and write latencies of the considered ReRAMs:

- ***SRAM-only:*** Baseline architecture similar the one shown in Figure 3.28. The main memory (96 kB PM and 64 kB DM) is composed of 6T SRAMs and is directly interfaced with the processing cores through crossbars, without page buffers.

- ***ReRAM-only:*** Baseline architecture with a single level of memory (96 kB PM and 64 kB DM) implemented entirely with ReRAMs and directly interfaced with the processing cores through crossbars, without page buffers.

- ***Hybrid-BC:*** This architecture is similar to the one proposed in [30], with the main memory (96 kB PM and 64 kB DM) implemented with ReRAMs and the page buffers (192 B P-PB and 256 B D-PB) realized using Standard Cell-based Memories (SCM). I have considered the best case for the ReRAM write energy of E_{BC} = 1.248 pJ/bit (cf. Section 3.3.3).

- ***Hybrid-WC:*** This configuration is the same as Hybrid-BC, with a worst-case ReRAM write energy of E_{WC} = 5.472 pJ/bit (cf. Section 3.3.3).

System power characterization

To estimate the power consumption of the system, a high-level cycle-accurate SystemC simulator of the entire platform was employed to derive the various required statistics relative to each component for the entire application runtime. The power numbers for the processing elements were derived from a post-place-and-route netlist of the system described in Figure 3.28, considering a SoA 28 nm low-power, high-k metal gate, Process Design Kit (PDK) technology under nominal supply conditions (V_{dd} = 1 V) at 300 K. These values are used in conjunction with the obtained simulator statistics for the considered application to derive the average power of the system.

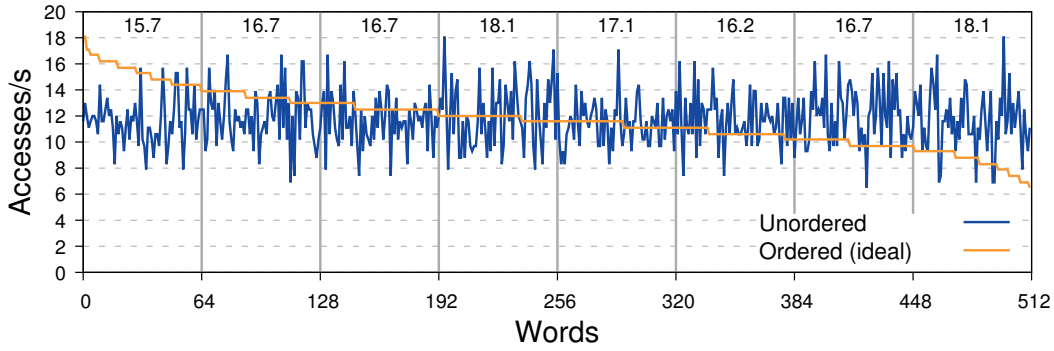


Figure 3.29 – Write accesses distribution over data words for CS. Vertical bars show the grouping of words into 64-word blocks, with the maximum number of writes to any word in each block. Also shown (orange) is the distribution of accesses if the application variables can be reordered.

Table 3.4 shows the values for dynamic energy consumption and leakage power used in this study for the different types of memories in each platform configuration. The parameters for the ReRAMs were derived using NVsim [169], with the programming conditions detailed in Section 3.3.3. For the SRAM main memory, they were calculated with the CACTI memory modeling tool [170]. In this study, it was assumed that the energy consumption per operation for the SCM page buffers is similar to that of an SRAM of equivalent size. This assumption is accurate for memories with small sizes (as in the case of the page buffers here).

3.3.4 Experimental results

3.3.4.1 Framework

This section details the adopted framework, explaining the studied BSP application, ReRAM reliability, and the lightweight block substitution mechanism that is employed to enhance the ReRAM lifetime.

Target bio-signal processing application

An 8-lead ECG input Compressed Sensing (CS) application (cf. Section 2.3.3.3) was used to evaluate the platform behavior with a real-life BSP application. CS performs the lossy encoding (50 %) of the input ECG signals, with the computation respective to each lead mapped on a different processing core. It exhibits a random memory access pattern typical of WBSN applications, that stresses traditional caching solutions. The input ECG signals were derived from the MIT-BIH Normal Sinus database [102], considering ECG windows of 1024 samples, acquired with a frequency of 500 samples/second. The entire execution time (acquisition and processing) for the CS application on the target multi-core WBSN platform for a window of samples takes 2.16 s.

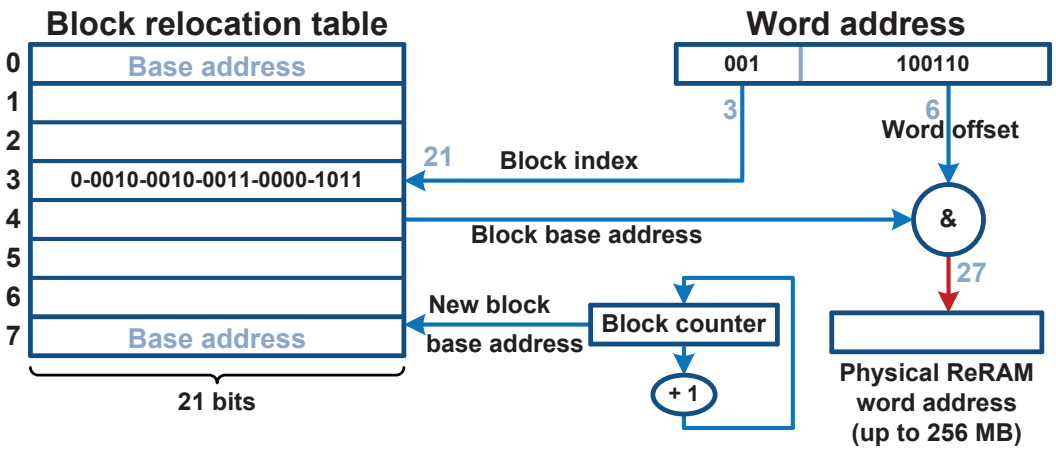


Figure 3.30 – The proposed ReRAM block relocation mechanism (maximum 256 MB addressable) embedded in the MMU.

ReRAM reliability and lightweight block substitution mechanism for enhanced durability

The main drawback of ReRAMs is their relatively low endurance (from 10^4 to 10^8 cycles [171] in academic foundries and $\approx 10^6$ cycles qualification in commercially available products [163]). This is less important when they are employed for storage, but becomes crucial when ReRAMs are used as a substitute of SRAMs for main memory, especially in WBSNs.

Contrarily to flash technologies that are block addressable, ReRAMs enable addressing granularity at the bit level, thereby moving the aging from blocks to the bits. As the data is written in ReRAM memories per word, I have considered the endurance at the level of individual words. Moreover, this characteristic enables the analysis of ReRAM aging, knowing the access pattern of the application, and permitting solutions that work at a small granularity. In the analyzed CS application, individual memory words can be identified that are written approximately 18 times per second. If a worst-case scenario is assumed in which a ReRAM word cell fails after 10 000 writes, then some words in the ReRAM will start to fail after roughly 555 s (less than 10 minutes). This is clearly an insufficient WBSN lifetime for any practical purpose.

Previous works have explored complex mechanisms to protect ReRAMs against permanent failures in high capacity NVMs [172] [173], whose study is out of the scope of this thesis. Herein, I use a lightweight energy-efficient mechanism to substitute failed memory words transparently for software applications.

The application data space was divided into small blocks, which are replaced when any of their words fails. Hence, there is a trade-off between management overhead and underexploited storage. In this study, I have divided the 512 data words of CS into 8 blocks with 64 words each. Figure 3.29 shows the distribution of writes to the ReRAM main memory for the CS application. Interestingly, if the data words correspond to small data objects, they can be reordered according to the number of write accesses to achieve more homogeneous blocks

and better storage exploitation. This is not the case for the CS application, which works over a single data array.

To resolve the issue of endurance-related failures, I propose the mechanism shown in Figure 3.30 to implement ReRAM block replacement. A small Block Relocation Table (BRT) with 8 entries holds the base address of each block. For every processor access to main data memory, the address is decomposed into a block index part and a word (offset) part. The block index is used to select an entry in the BRT that supplies the current base address of the block in the ReRAM, which is then concatenated with the word offset to create the final ReRAM address.

When a word in a block fails, the complete block is relocated to a different position in the ReRAM. Consequently, the block counter is increased by one and its value is used as the new starting address for that block. This is an atomic operation and the block allocation is sequential, which ensures that the new starting address is unused. The BRT and the block counter were implemented as 21-bit registers (with a total size of 189 bits) and their read energy for every access to the ReRAM were also taken into account. I have assumed that the BRT remains always active (to reduce the number of writes to ReRAM) and account for its leakage power as well.

3.3.4.2 Obtained results

This section showcases the results obtained, in particular: studying ways for memory hierarchy optimization for BSP applications, power consumption evaluation, performance evaluation of low access locality applications, and an analysis of the device lifetime based on ReRAM endurance.

Memory hierarchy optimization for WBSN applications

Page buffers work adequately for algorithms with high access locality. Unfortunately, several key applications in the WBSN domain, such as CS, exhibit markedly random access patterns. In those cases, caching may actually produce write amplification (i.e., when the actual amount of information physically written to the main memory is a multiple of the logical amount intended to be written) because whole lines are replaced after a few words are used. In the proposed architecture, each page buffer eviction writes 8 words (128 bits) to memory, even if just one word was modified. With SRAM or DRAM-based architectures, this is mainly an issue of wasted energy (and performance).

However, due to their higher write current, in ReRAM-based memories writing 128 bits in parallel may be unfeasible. For example, with a 100 μA programming current per bit, a 128-bit parallel write requires a current of 12.8 mA. Such a current would require strong modifications to the macro memory design to avoid reductions in both chip and battery reliability. A possible solution would be multi-cycle write-backs, e.g., a 4-cycle latency per page buffer write-back for a maximum current of 3.2 mA. Nevertheless, if only a few words are modified, ReRAMs still

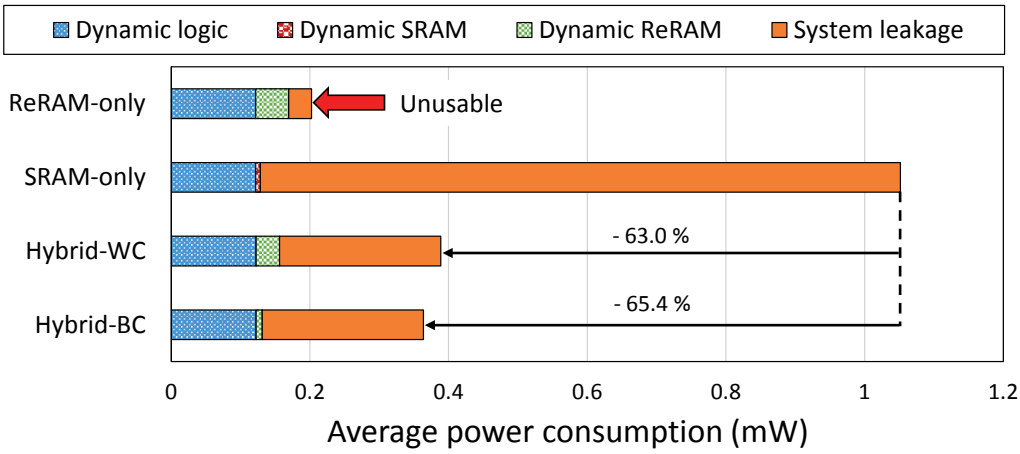


Figure 3.31 – Average power of the system components at 1 V supply.

consume some energy internally to avoid overwriting bits which are already in the intended resistance state.

To better cope with such complex access patterns, I extend the page buffers' *dirty bits* to cover individual words, at the cost of 128 extra bits. During write-backs, the MMU checks the state of the dirty bits for word masking before sending requests to the ReRAM main memory.

Power and energy consumption analysis

Figure 3.31 showcases the system's average power requirements in the different considered configurations. *Dynamic logic* accounts for the dynamic power of processing components like cores, crossbars, clock tree, etc., along with that of the page buffers and the BRT, when applicable. *Dynamic SRAM* and *Dynamic ReRAM* are the dynamic power of the SRAM and ReRAM main memories, respectively. Finally, *System leakage* depicts the combined leakage power of the entire system.

The SRAM-only configuration has the highest power requirement, with leakage accounting for the majority of it. Although the ReRAM-only platform requires more dynamic power compared to the SRAM-only system, it decreases the leakage power by over 95 % by power-gating the entire platform when the cores are idle. Even if these results make the ReRAM-only configuration desirable, the ReRAM main memory is written a large number of times due to the absence of page buffers, resulting in numerous premature stuck-at failures. Thus, this option is unusable in practice at current ReRAM endurance rates. Finally, the evaluated hybrid architectures still diminish substantially the leakage power, even when the page buffers and the BRT are taken into account: although dynamic power is marginally higher than in the SRAM-only configuration, significant overall power reductions of 63.0 % and 65.4 % are achieved, in the worst case and best case, respectively.

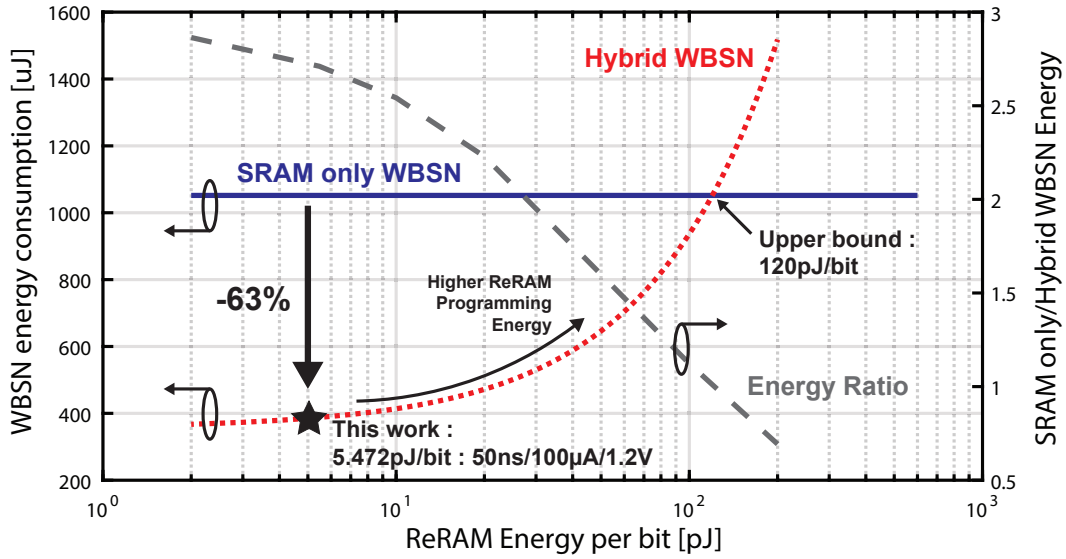


Figure 3.32 – WBSN energy consumption embedding SRAM (blue) or ReRAM (red) against ReRAM programming energy. Energy ratio between SRAM and ReRAM based WBSNs in gray.

Figure 3.32 shows the evolution of the energy consumed in two WBSN configurations: SRAM in blue and ReRAM with page buffers (hybrid) in red. To ease the reading, the energy ratio between these two configurations is also shown (gray dotted line). With the assumptions taken in this work (i.e., the worst case programming energy is 5.472 pJ/bit), the proposed system consumes 63 % less energy than the SRAM-based one. However, for several reasons (e.g., process variability, temperature, retention constraints or use of complex iterative write strategies), the average programming energy of the ReRAM may be higher or lower than the considered values. To generalize this study, I have determined the energy consumption of the proposed architecture for a wide range of ReRAM programming energies. Obtained results show that the proposed architecture is advantageous compared to the SRAM-based approach until the ReRAM programming energy reaches 120 pJ/bit, which corresponds to a 1 µs programming pulse with a 100 µA current at 1.2 V. Thereby, the proposed solution is suitable even for high programming energy ReRAM technologies.

Low access locality performance evaluation

The architectural optimizations introduced in Section 3.3.4.1 to improve page buffer performance in cases of low access locality are successful for the considered CS application. Without proper page buffer sizes, large amounts of writes would be made to the NVM, suffering a 6.2x write amplification to main (ReRAM) memory. The reduction in the number of words written to the ReRAM (from 874 312 to 140 729 writes) can be used to optimize the structure of the bus between the page buffers and the ReRAM, and to subsequently reduce the energy needed by

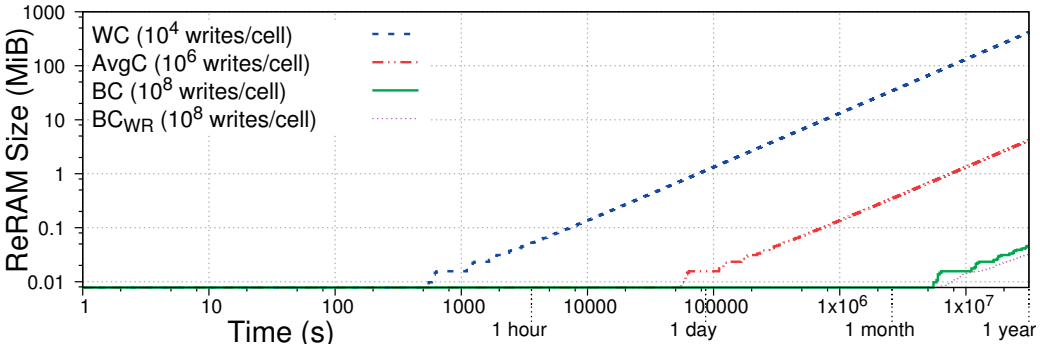


Figure 3.33 – ReRAM over-provisioning requirements to achieve up to one year of system lifetime, depending on the number of writes to a word before it fails permanently.

the ReRAM itself to test the previous values of the affected array cells. Finally, out of the words sent to the ReRAM for writing, the percentage of bits actually changed over the course of one sampling window is found to be only 39.5 %.

Analysis of device lifetime based on ReRAM endurance

This section evaluates the additional ReRAM capacity needed to be included in the system to ensure long-term WBSN operation. Since the reliability conditions of ReRAMs depend on several factors such as manufacturing process, physical size or programming current, I have selected three possible endurance ranges: memory cells fail permanently after 10^4 writes (*Worst Case*), after 10^6 writes (*Average Case*), or after 10^8 writes (*Best Case*), which cover a wide scope of ReRAM endurance.

Using the average number of writes per second for each block obtained in Section 3.3.4.1, Equation 3.3 calculates the number of blocks needed along time (hence, the required ReRAM capacity) per core:

$$Blocks(t) = \sum_{i=1}^{Blocks} \left\lceil \frac{t \times A_i}{W} \right\rceil \tag{3.3}$$

where A_i is the number of accesses per second to block i, and W is the estimated number of writes to a cell before failure.

Figure 3.33 shows the minimum ReRAM capacity required to enable up to one year of WBSN operation. In the *Best Case* (BC in the figure), the system can work for one year using 47 KiB of ReRAM (considering the CS application). As the endurance of the ReRAM is decreased, the system requires progressively more memory to remain operative during the same amount of time. Thus, in the *Average Case* (AC), it will require 4.2 MiB of total ReRAM capacity to

operate for one year. Whereas, in the *Worst Case* (WC), it will require up to 417 MiB for the same period of operation. Figure 3.33 also shows that the ReRAM capacity required with a hypothetical optimal block-replacement strategy working on individual words is 34 KiB (*Best Case, Word-Replacement*, BC_{WR}), providing a comparison for the lightweight solution.

As a conclusion, further optimization on the construction and programming of ReRAMs to enhance their durability are crucial to profit from their benefits on leakage reduction in the design of reliable embedded devices. Moreover, such improvements would also entail a potential reduction in costs as ReRAMs typically present an area reduction factor of 4x compared to SRAMs.

3.4 Summary of contributions

In the research work presented in this chapter, I have introduced heterogeneous architectures for the efficient execution of BSP applications: a CGRA-based accelerator at the architectural level and emerging ReRAM technologies at the memory level. Additionally, I have explored the application of inexact-enabled relaxed-reliability computation on top of the proposed CGRA-based heterogeneous BSP platform. The contributions of this chapter can be broadly classified into three parts, as described in the following.

In Section 3.2.2.3, I have integrated a time- and space-shared-CGRA-based accelerator for BSP, in conjunction with traditional multi-core WBSNs, in order to execute computation-intensive kernels present in BSP algorithms more efficiently. To this end, I have first implemented a CGRA architecture adapted to low-power BSP, detailing its building blocks and execution methods. Then, I have further optimized the CGRA to include multiple DPs, in order to support SIMD execution. In this way it is possible to converge multiple calls of the same kernel from different cores to be executed on multiple DPs that are governed by the same shared control logic. The most notable contributions in this research path are the design of reconfigurable architectures that act as shared resources and support SIMD, and the methods developed to map the various selected kernels on the CGRA (derived from modulo scheduling) along with their execution, addressing constraints on both the hardware and the software sides. Following this step, I have tested the proposed heterogeneous scheme by employing a wide range of BSP applications, achieving kernel speedups of up to 11.3x and significant overall system energy reductions of up to 37.2 %, with acceptable additional area overheads.

Then in Section 3.2.3, I optimized the CGRA design further to efficiently execute typical BSP kernels, that can parallelize kernels even from a single core, by employing an interleaved-datapaths (i-DPs) strategy. In this approach, the execution of kernels are divided over the input data size and each resulting sub-part is processed by a different DP. In this way, the i-DPs parallelize the execution of kernels, while sharing the more power-hungry control logic. As in the previous part, herein I have also explored methods to map the selected kernels efficiently on the CGRA, considering the skewing of instructions issued in the employed DPs and constraints on the architecture and application levels. The results of my experiments

show substantial reductions in the kernel execution time compared to a single-DP CGRA. By employing the i-DPs CGRA, it is possible to improve the energy efficiency of the entire system by up to 33 % over a CGRA featuring just 1 DP, incurring limited area penalty.

In Section 3.2.4, I have explored the joint application of CGRA-based acceleration and relaxed-reliability computation in the BSP domain. I have shown that by relaxing the accuracy requirements of the system, by allowing errors in the memory subsystem due to voltage over-scaling to propagate to both processors and the CGRA, it is possible to achieve additive gains in the energy efficiency of WBSNs. In particular, I have studied the effect of voltage supply droops and fabrication process variation on SRAM memory cells, observing the manifestation of memory failures on the application-visible outcomes. To this end, I have classified the errors depending upon their severity as seen on the system output, consequently employing a lightweight error monitoring strategy. Experimental results highlight that as much as 70.7 % reduction in the energy budget can be achieved by using the proposed reconfigurable system with relaxed-reliability requirements over a traditional multi-core-only WBSN, while at the same time ensuring a high quality of the produced output.

While the optimization techniques discussed in the previous paragraphs are highly instrumental in improving the (mainly dynamic) energy efficiency of WBSNs, however with advancements in technology, leakage power is emerging as the dominant contributor. To tackle this issue, in Section 3.3, I have explored the integration of emerging non-volatile ReRAM-based memories (with negligible leakage) in traditional WBSNs as the main storage unit, with volatile page buffers as a first level storage system. This approach is aimed at lowering the leakage component drastically, while at the same time reducing the number of accesses to the ReRAM main memory as they start to fail after a relatively low number of writes. Performed experiments showcase that by effectively applying power management schemes during typically long idle periods prevalent in BSP applications, it is possible to achieve significant system-level energy savings of up to 65.4 %, compared to a WBSN with SRAM-based memory. To counter the endurance effects, I have introduced a lightweight block replacement strategy to improve the ReRAM lifetime, studying the trade-offs between the device lifetime and the additional ReRAM memory required to ensure that.

4 Conclusions and Future Work

To conclude this thesis, in this chapter, I list the main conclusions drawn from my research work and highlight the contributions that it has made to the State-of-the-Art (SoA) of low-power biomedical devices. In addition, I perform an exploration of the research directions possible for future work based on the results reported in this thesis.

4.1 Summary and contributions

In this thesis, I have proposed a set of hardware- and software-based co-design techniques that aim to improve the energy efficiency of low-power bio-medical devices. To tackle the problems of energy bottlenecks persistent in SoA Wireless Body Sensor Nodes (WBSN), I have explored various strategies to optimize the power consumption of Bio-Signal Processing (BSP) platforms. To this end, first, I studied the benefits of voltage over-scaling to ultra-low ranges, while relaxing the processing accuracy of WBSN platforms against failures ensuing as a result. To this end, I have proposed a range of hardware- and software-based counter measures. In parallel, I have also investigated the application of CGRA-based reconfigurable architectures in the field of BSP, aimed at improving the energy efficiency of WBSNs by optimizing the processing of computational kernels. Finally, I have studied the possibilities and issues in the integration of emerging Resistive Random Access Memory (ReRAM)-based memories in WBSNs, proposing lightweight strategies to maximize the device lifetime. The main contributions of this thesis are discussed in more details in the following.

Relaxed-reliability computation for bio-signal processing

In Chapter 2, I have proposed two complementary strategies to counter memory errors occurring as a result of voltage over-scaling:

In Section 2.2, I have presented methods which tackle memory errors by maximizing their mitigation before they are used by the processing cores. To this end, I have shown that a significance-based protection scheme can effectively reduce the energy consumption of WBSN

memories. My experiments have shown that, by adopting different correctness guaranties in words and bits of varying significance from a BSP perspective, the proposed approach can effectively reduce the energy overhead implicit in data protection, while keeping the output quality degradation within permissible bounds. Experimental results have highlighted that the proposed heterogeneous protection scheme reduces the energy budget of the data memory by up to 20 %, bringing system-wide savings of up to 5.2 %, while tolerating high error rates at reduced supplies.

Then in Section 2.3, I have applied an orthogonal approach where no dedicated memory protection scheme is used, while WBSNs are subjected to ultra-low voltage supplies in presence of runtime droops and process variations. Instead, by allowing errors emanating from the memories to spread to the entire system, here I have evaluated their criticality by studying their effect on the output quality, proposing lightweight error monitoring strategies to counter them. Proving that 100 % correctness guarantee is not required to avoid permanent system-level failures, especially in the BSP domain, I have shown that the proposed mitigation strategies are highly effective. Experimental results have shown that up to 24 % overall energy savings at the system level can be obtained, compared to an equivalent failure-free alternative.

From these observations, it can be concluded that inexactness-based relaxed-reliability computation can be successfully sustained in the BSP domain, resulting in significant energy savings, while requiring limited additional overhead to ensure acceptable computation.

Publications:

The work on significance-based memory protection scheme resulted in two publications: first at the *IEEE Annual Symposium on VLSI (ISVLSI)* [174] and an extension of the work to the entire WBSN system was published in the *IET Computers & Digital Techniques* journal [175]. The work on relaxed-reliability computation with error management was published in the *ACM Transactions on Embedded Computing Systems* journal [100].

Heterogeneous architectures for ultra-low power WBSN platforms

In Chapter 3, I have explored the integration of heterogeneous architectures in traditional WBSNs to improve their energy efficiency, from two different perspectives:

In Section 3.2, I have explored the application of Coarse Grain Reconfigurable Array (CGRA) architectures in coalition with traditional multi-core-based WBSN platforms. To this end, I have developed different versions of CGRA (SIMD and i-DPs) that are efficient in executing BSP computational kernels, with SIMD CGRA specialized for simultaneous multiple kernel calls, and i-DPs CGRA optimizing the execution of kernels with non-SIMD calls. By effectively mapping and executing the selected kernels on the CGRA, it is possible to achieve system energy savings of up to 37.2 % for SIMD CGRA and 33 % for the i-DPs CGRA, over a traditional 1 DP CGRA. Borrowing the ideas from Chapter 2, herein I have also explored the application

of relaxed-reliability computation along with CGRA acceleration for BSP, to jointly leverage the benefits derived from each scheme. Experimental results highlight that as much as 70.7 % reduction in the energy budget can be achieved by using the proposed reconfigurable system with relaxed reliability requirements over a traditional multi-core-only WBSN, while at the same time ensuring a high quality of the produced outputs.

To tackle the issue of leakage dominating the power consumption of predominantly *idle* WBSNs, in Section 3.3, I have analyzed the use of emerging non-volatile ReRAM memories in traditional BSP platforms. By taking advantage of the negligible leakage of ReRAMs, it is possible to cut the leakage power consumption significantly, resulting in system-level energy savings of up to 65.4 % compared to a WBSN with SRAM-based memory. Furthermore, I have introduced a lightweight block replacement strategy to improve the ReRAM lifetime, studying the trade-offs between the device lifetime and the additional ReRAM memory required to ensure that.

Experimental results obtained in this chapter show that heterogeneous CGRA-based architectures are favorable solutions for further increasing the energy efficiency of WBSNs. Moreover, the integration of ReRAM in WBSNs is highly effective in reducing their leakage power, thereby lowering their energy budgets substantially.

Publications:

The work on CGRA-based heterogeneous BSP architectures has resulted in three publications: the study on the basic single-DP CGRA was presented at the *Biomedical Circuits and Systems (BioCAS)* conference [150] and the extension of this work to include the SIMD CGRA has been published in the *IEEE Transactions on Circuits and Systems I* journal [25]. In addition, my work on the i-DPs CGRA has been published in the *IEEE Embedded Systems Letters (ESL)* journal [176]. The research combining CGRA acceleration and relaxed-reliability computation in the BSP domain was accepted in *The International Symposium on Circuits and Systems (ISCAS)* [177]. Finally, the work involving the integration of ReRAM memories in BSP architectures has been accepted as “Work in progress” the *Design Automation Conference (DAC)*.

4.2 Future Work

The research findings that have been showcased in this thesis open the door for future research directions that can be pursued in this field. A set of short-term developments that continues the proposed approaches to achieve higher energy efficiency in WBSNs is summarized in the following:

- **Relaxed-reliability computation in Interleaved Datapaths (i-DPs) CGRA**

  In this thesis, I have studied the joint application of relaxed-reliability computation and CGRA-based acceleration in WBSNs, focusing on an SIMD CGRA as the case study.

However, this technique can be extended to the CGRA with i-DPs also. In this scenario, similar to the presented research, the corrupted data due to memory errors are also processed by the CGRA, in addition to the multi-processors. This study will potentially expose further the error-tolerant characteristics of BSP applications, presenting the opportunity to study the trade-offs between the improvement of the energy efficiency and the degradation of the output quality. In this way, it will be possible to chose an optimal voltage of operation for BSP application kernels that process lengthy input data, or are called in a non-SIMD mode.

- **Inexact logic components in the CGRA and processors**

  In this thesis, I have explored the application of relaxed-reliability computation in CGRA-based heterogeneous WBSN architectures, focusing on the inexactness resulting out of memory errors when operating at ultra-low voltages. However, further techniques imposing inexactness computation in this domain can be explored. One such approach is to use inexact computing elements, such as inexact adders and multipliers in the CGRA cells and for the processors in the multi-core architecture. A parallel strategy is to probabilistically prune parts of some of the adders or multipliers in the CGRA or processors, depending upon their usage statistics [62]. Since a major part of the computing logic is composed of adders and multipliers in the processing cores as well as in the CGRA, this method can potentially further reduce their energy budgets significantly. A final step in such a study would be to estimate the effects of using the proposed inexact logic elements in the WBSN architecture on the application outputs, exploring the ensuing trade-offs.

- **Automatic kernel mapping and CGRA selection**

  In the research work presented in Chapter 3, a manual approach has been employed for the mapping of the selected computation-intensive kernels of BSP applications onto the CGRA accelerator [144]. In the future, this step can be automated, thereby creating a framework where entire applications can be smoothly executed on a CGRA-based heterogeneous platform. In this regard, previous works in the literature exist, that have created methodologies for kernel selection [178] and automatic generation of the control flow in the kernel code [179] [180]. However, they are not optimized for heterogeneous reconfigurable platforms like the one proposed in this thesis. The next step in advancing this research can be to use these techniques to automatically select the kernels from BSP applications and map them onto the CGRA. For the automatic kernel-selection process, the hardware and software constraints enforced by the architecture of the employed CGRA also need to be taken into account. Then, using a derivative of modulo scheduling [181], the kernels can be mapped efficiently onto the reconfigurable mesh, optimizing the number of CGRA Reconfigurable Cells (RC) and columns that are used. In addition, when a kernel mapping is obtained and the dependencies have been resolved, the generation of the corresponding bit-streams can also be automated using a specific lightweight compiler. This work will enable the study of trade-offs in the energy

consumed and the speedup achieved in the execution of kernels while employing the automatic mapping scheme, compared to a manual approach as adopted in this thesis.

In the work presented in this thesis, I have explored various CGRA architectures, the use of each of which is especially beneficial for certain types of application kernels. While in this study, the desired CGRA architecture has to be selected manually after profiling the applications, in the future this step can be automated as well. In particular, it is needed to extract various characteristics relative to each application kernel (such as input vector length, number of SIMD calls, number of calls by a single core). These parameters, in turn, will be useful in determining the target CGRA. For example, if a kernel has high number of simultaneous calls in parallel, its execution will be most efficient if mapped on an SIMD CGRA; while if it processes lengthy input vectors in a non-SIMD mode, it can be mapped on the i-DPs CGRA. In addition, the CGRA dimensions (number of RCs) and the related hardware support can be made flexible. Therefore, depending on the application domain, it will be possible to chose the optimum CGRA parameters (like SIMD width, number of columns, number of configuration registers, etc.). This methodology promises to open new dimensions for studying the trade-offs involved in performing the selection of the optimum CGRA architecture and the related parameters.

- **Fully-connected CGRA for bio-signal processing**

  In this thesis, the various CGRA architectures that have been employed feature a neighbor-to-neighbor connection among the RCs, arranged in a 4×4 torus configuration. While this architecture is sufficient for mapping the application kernels that have been considered, their execution can be further optimized. One such way is to adopt a CGRA design where each RC is connected with all other RCs in the mesh. This condition removes the constraints on kernel mapping imposed by the CGRA mesh used in this work, thereby making the reconfigurability even higher. In this way, there will no longer be unused RCs in an employed column, and power-gating of RCs can be done on the granularity of each cell. This work could involve the creation of a centralized register file, which each RC can access, that will eliminate the requirement of register files per RC, thereby making the kernel mapping process much simpler by enabling data-sharing. In this case, the program counters for configuring RCs and executing the kernels have to be reworked. Since the operational granularity will be per RC, a set of configuration and execution program counters need to be associated with each of them.

In addition, and stemming from the proposed work, some long-term research lines, which could potentially reduce the energy requirements and increase the reliability of next generation WBSNs, are discussed the following:

- **Extension of relaxed-reliability computation to other application domains**

  While in this thesis the application of relaxed-reliability computation has been limited to BSP applications, it can be potentially applied to other application domains, where

energy efficiency is critical and a 100 % correctness guarantee is not required. As a first step, the study of the application of relaxed-reliability computation with error management can be evaluated against further BSP applications. An ensuing energy vs Quality-of-Service (QoS) trade-off study will consolidate the effectiveness of such a strategy in the entire BSP domain.

Another application domain where the relaxed-reliability computation strategies studied in this thesis can be applied is that of neural networks. This is an important emerging research domain, where the improvement of energy efficiency can be highly beneficial. Neural network applications are also inherently error tolerant, as they involve rigorous training phases and widespread data quantization. These properties motivate the application of inexact computation in this domain. In particular, first, the application relaxed-reliability computation with error management can be explored, studying the ensuing energy/QoS trade-offs. If this strategy does not prove to be efficient enough from the output degradation point-of-view, then a more elaborate one involving a data significance-based heterogeneous memory protection scheme can also be explored.

- **Data bandwidth improvement between the CGRA and data memory**

  Although the Direct Memory Access (DMA) energy overheads are accounted for in this thesis work, an extension of this study concerns the design and optimization of the DMA block itself. In the future, a DMA block can be implemented in hardware that can efficiently manage the transfer of data between the data memory and the CGRA, considering the constraints imposed by the CGRA and the WBSN platform design on the bandwidth of data transfer. Conversely, it can also be possible to improve the data transfer bandwidth. This work calls for new design strategies for the crossbar, that can manage more efficiently the data transfers between the memories and the processing cores/CGRA. This study can expose potential trade-offs in the crossbar design, as they can increase the energy consumption of the system while increasing the data bandwidth.

- **Exploration of backing up of WBSN memory to ReRAM**

  In the work presented in the Section 3.3 of this thesis, I have employed an architecture that features page buffers to reduce the number of accesses to the ReRAM main memory. While this approach is highly effective in improving ReRAM lifetime for applications with high access locality, it is not the most optimal solution for the ones with low access locality. In the latter case, a lot of misses occur in the page buffers, thereby necessitating a high number of data transfers between the ReRAM and the page buffers. This effect may even in some cases negate the endurance improvement achieved by using page buffers, also adversely impacting the ReRAM endurance. An alternative approach for applications exhibiting these characteristics is to abandon the page buffers, thereby backing up the entire SRAM instruction and data memories to a ReRAM block. As there are very few sleep periods for some BSP applications, this option may actually be more beneficial for ReRAM endurance than the one proposed in this thesis. Additionally, the extra ReRAM blocks needed to back the WBSN memories up is expected to be significantly smaller than what was required in the presented research.

A Reliability analysis of ReRAM-based heterogeneous WBSN memory featuring SRAM page buffers

In Section 3.3, I have presented a study on emerging ReRAM-based WBSNs, considering that the Page Buffers (PB) in the system are implemented with Standard Memory Cells (SCM). This assumption is reasonable given the small size of the PB (192 B for program PB and 256 B for data PB), where the energy consumption of SCMs are comparable to SRAMs. Nevertheless, I have performed the study of the effects of implementing the PBs with 8-transistor SRAMs, which is detailed in this appendix.

First, I have studied the defects of voltage scaling in the PBs, considering runtime droops and fabrication variabilities, as in Section 2.3. Given the small size of the PBs, I was able to make an exhaustive study of the manifestation of bit-flips at each memory cell into application visible effects. Figure A.1 compares the error distribution in the case of an SRAM-only memory without PBs and the proposed heterogeneous one featuring ReRAM main memory with SRAM PBs, weighted with the access frequency of the memory words. It can be seen that there is a significant increase in the number of silent errors, i.e., the ones which do not have any application-visible effect. This is because the probability of error is reduced according to

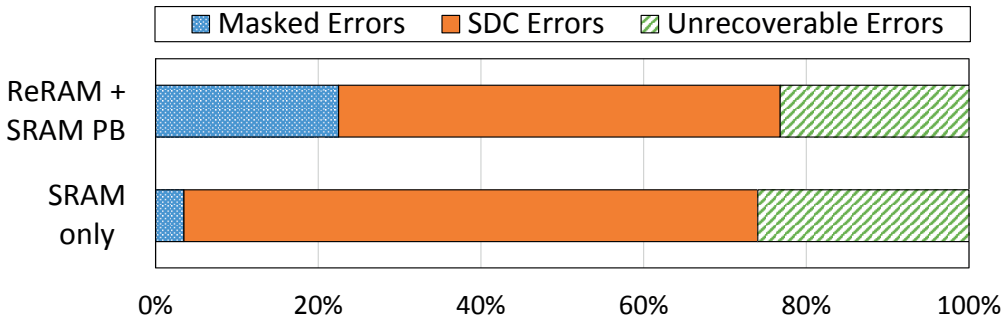


Figure A.1 – Run-time error profile comparison of the studied architecture against an SRAM-only one, considering the access frequency of memory words.

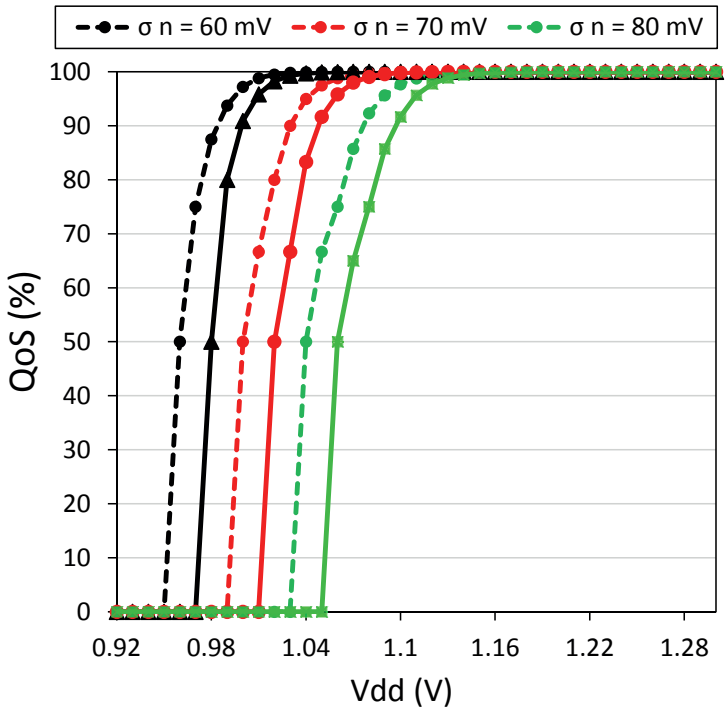


Figure A.2 – Comparison of the Quality-of-Service (QoS) evolution for the two architectures for different voltage supply values and droop variations. The dotted line represents the condition with SRAM page buffers.

the substantially reduced size of the SRAM-based memory (only PB compared to the entire main memory), and also because these cells have a high access frequency compared to the SRAM-only case. Nevertheless, the SDC errors (cf. Section 2.3.2.2) still form a majority, with unrecoverable errors accounting for a significant part. This can be explained as there maybe bit-flips in cells producing SDC or unrecoverable errors with a high number of accesses as in the SRAM-only platform.

Secondly, I have also investigated the effects of these errors in the overall Quality-of-Service (QoS) of the system, which is defined in the same way as in Section 2.3.3.3, as the ratio of correctly output ECG windows to the total number of processed windows. An experimental setup similar to the one presented in Section 2.3.3.3 is also assumed for this study, considering Compressed Sensing (CS) as the target BSP application [13]. Figure A.2 analyzes the QoS of the system at varying supply voltages, with different voltage droop variations. It can be seen that a QoS of 90 % can be ensured with a supply voltage of 1.02 V for the heterogeneous system, compared to 1.05 V for the SRAM-only platform (considering a voltage droop variance of 70 mV), thereby enabling a voltage reduction of up to 3 %. Similarly, for ensuring a QoS of 99 %, the effective voltage reduction that can be achieved is up to 2 %, thereby improving the energy efficiency of the platform by an additional 4 % over what has been accomplished as shown in Section 2.3.3.3.

Bibliography

[1] The World Health Organization, "Global status report on non-communicable diseases, 2014." http://apps.who.int/iris/bitstream/handle/10665/148114/9789241564854_eng?sequence=1, 2014.

[2] The World Health Organization, "Non-communicable diseases." http://www.who.int/mediacentre/factsheets/fs355/en/, 2016.

[3] The United Nations, "World Population Prospects." https://esa.un.org/unpd/wpp/Publications/Files/WPP2017_KeyFindings.pdf, 2017.

[4] Organisation for Economic Co-operation and Development (OECD), "Health-care cost." https://stats.oecd.org/Index.aspx?DataSetCode=SHA, 2018.

[5] MEP, "Cardiovascular diseases facts and figures." http://www.mepheartgroup.eu/index.php/facts-a-figures, 2015.

[6] Y. Hao and R. Foster, "Wireless body sensor networks for health-monitoring applications," *Physiological Measurement*, vol. 29, no. 11, p. R27, 2008.

[7] E. Jovanov, A. Milenkovic, C. Otto, and P. C. de Groen, "A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation," *Journal of NeuroEngineering and Rehabilitation*, vol. 2, p. 6, Mar 2005.

[8] SmartCardia. http://www.smartcardia.com/.

[9] P. Bonato, "Wearable sensors and systems," *IEEE Engineering in Medicine and Biology Magazine*, vol. 29, pp. 25–36, May 2010.

[10] T. Berset, I. Romero, A. Young, and J. Penders, "Robust heart rhythm calculation and respiration rate estimation in ambulatory ecg monitoring," in *Proceedings of 2012 IEEE-EMBS International Conference on Biomedical and Health Informatics*, pp. 400–403, Jan 2012.

[11] F. Rincón, J. Recas, N. Khaled, and D. Atienza, "Development and evaluation of multilead wavelet-based ecg delineation algorithms for embedded wireless sensor nodes," *IEEE Transactions on Information Technology in Biomedicine*, vol. 15, pp. 854–863, Nov 2011.

[12] R. Braojos, H. Mamaghanian, A. Dias, G. Ansaloni, D. Atienza, F. J. Rincón, and S. Murali, "Ultra-low power design of wearable cardiac monitoring systems," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, June 2014.

[13] H. Mamaghanian, N. Khaled, D. Atienza, and P. Vandergheynst, "Real-time compressed sensing-based electrocardiogram compression on energy-constrained wireless body sensors," in *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, pp. 1744–1747, May 2011.

[14] Y. Sun, K. L. Chan, and S. M. Krishnan, "Ecg signal conditioning by morphological filtering," *Computers in Biology and Medicine*, vol. 32, no. 6, pp. 465 – 479, 2002.

[15] E. Imah, W. Jatmiko, and T. Basaruddin, "Adaptive multilayer generalized learning vector quantization (amglvq) as new algorithm with integrating feature extraction and classification for arrhythmia heartbeats classification," pp. 150–155, 10 2012.

[16] A. Cohen and J. Kovacevic, "Wavelets: the mathematical background," *Proceedings of the IEEE*, vol. 84, pp. 514–522, April 1996.

[17] D. C. Daly and A. P. Chandrakasan, "A 6-bit, 0.2 v to 0.9 v highly digital flash adc with comparator redundancy," *IEEE Journal of Solid-State Circuits*, vol. 44, pp. 3030–3038, Nov 2009.

[18] E. Nemati, M. J. Deen, and T. Mondal, "A wireless wearable ecg sensor for long-term applications," *IEEE Communications Magazine*, vol. 50, pp. 36–43, January 2012.

[19] Texas Instruments, ""Measuring Bluetooth Low Energy Power Consumption." http://www.ti.com/lit/an/swra347a/swra347a.pdf, 2016.

[20] F. Zhang, J. Holleman, and B. P. Otis, "Design of ultra-low power biopotential amplifiers for biosignal acquisition applications," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 6, pp. 344–355, Aug 2012.

[21] R. Braojos, A. Dogan, I. Beretta, G. Ansaloni, and D. Atienza, "Hardware/software approach for code synchronization in low-power multi-core sensor nodes," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1–6, March 2014.

[22] S. D. Carlo, A. Savino, A. Scionti, and P. Prinetto, "Influence of parasitic capacitance variations on 65 nm and 32 nm predictive technology model sram core-cells," in *2008 17th Asian Test Symposium*, pp. 411–416, Nov 2008.

[23] A. Y. Dogan, D. Atienza, A. Burg, I. Loi, and L. Benini, "Power/performance exploration of single-core and multi-core processor approaches for biomedical signal processing," in *Integrated Circuit and System Design. Power and Timing Modeling, Optimization, and Simulation* (J. L. Ayala, B. García-Cámara, M. Prieto, M. Ruggiero, and G. Sicard, eds.), (Berlin, Heidelberg), pp. 102–111, Springer Berlin Heidelberg, 2011.

[24] R. Braojos, D. Bortolotti, A. Bartolini, G. Ansaloni, L. Benini, and D. Atienza, "A synchronization-based hybrid-memory multi-core architecture for energy-efficient biomedical signal processing," *IEEE Transactions on Computers*, vol. 66, pp. 575–585, April 2017.

[25] L. Duch, S. Basu, R. Braojos, G. Ansaloni, L. Pozzi, and D. Atienza, "Heal-wear: An ultra-low power heterogeneous system for bio-signal analysis," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, pp. 2448–2461, Sept 2017.

[26] X. Zhang, H. Jiang, L. Zhang, C. Zhang, Z. Wang, and X. Chen, "An energy-efficient asic for wireless body sensor networks in medical applications," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 4, pp. 11–18, Feb 2010.

[27] R. H. Dennard, H. Robert, V. Rideout, E. Bassous, and A. LeBlanc, "Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 9, pp. 256–268, October 1974.

[28] A. Wang and A. Chandrakasan, "A 180-mv subthreshold fft processor using a minimum energy design methodology," *IEEE Journal of Solid-State Circuits*, vol. 40, pp. 310–319, Jan 2005.

[29] M. Ashouei, J. Hulzink, M. Konijnenburg, J. Zhou, F. Duarte, A. Breeschoten, J. Huisken, J. Stuyt, H. de Groot, F. Barat, J. David, and J. V. Ginderdeuren, "A voltage-scalable biomedical signal processor running ecg using 13pj/cycle at 1mhz and 0.4v," in *2011 IEEE International Solid-State Circuits Conference*, pp. 332–334, Feb 2011.

[30] R. Braojos, D. Atienza, M. M. S. Aly, T. F. Wu, H. S. P. Wong, S. Mitra, and G. Ansaloni, "Nano-engineered architectures for ultra-low power wireless body sensor nodes," in *2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pp. 1–10, Oct 2016.

[31] T. Simunic, L. Benini, and G. D. Micheli, "Cycle-accurate simulation of energy consumption in embedded systems," in *Proceedings 1999 Design Automation Conference (Cat. No. 99CH36361)*, pp. 867–872, June 1999.

[32] S. Ganapathy, G. Karakonstantis, R. Canal, and A. P. Burg, "Variability-aware design space exploration of embedded memories," in *2014 IEEE 28th Convention of Electrical Electronics Engineers in Israel (IEEEI)*, pp. 1–5, Dec 2014.

[33] M. M. Sabry, G. Karakonstantis, D. Atienza, and A. Burg, "Design of energy efficient and dependable health monitoring systems under unreliable nanometer technologies," in *Proceedings of the 7th International Conference on Body Area Networks*, BodyNets '12, (ICST, Brussels, Belgium, Belgium), pp. 52–58, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2012.

[34] N. Verma and A. P. Chandrakasan, "A 256 kb 65 nm 8t subthreshold sram employing sense-amplifier redundancy," *IEEE Journal of Solid-State Circuits*, vol. 43, pp. 141–149, Jan 2008.

[35] D. Bortolotti, H. Mamaghanian, A. Bartolini, M. Ashouei, J. Stuijt, D. Atienza, P. Vandergheynst, and L. Benini, "Approximate compressed sensing: Ultra-low power biosignal processing via aggressive voltage scaling on a hybrid memory multi-core processor," in *2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 45–50, Aug 2014.

[36] T. Opthof, "The normal range and determinants of the intrinsic heart rate in man.," *Cardiovascular research*, vol. 45 1, pp. 173–6, 1970.

[37] H.-S. P. Wong, C. Ahn, J. Cao, H.-Y. Chen, S. B. Eryilmaz, S. W. Fong, J. A. Incorvia, Z. Jiang, H. Li, C. Neumann, K. Okabe, S. Qin, J. Sohn, Y. Wu, S. Yu, and X. Zheng, "Stanford Memory Trends." Accessed November 20, 2015.

[38] W. Massagram, N. Hafner, M. Chen, L. Macchiarulo, V. M. Lubecke, and O. Boric-Lubecke, "Digital heart-rate variability parameter monitoring and assessment asic," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 4, pp. 19–26, Feb 2010.

[39] X. Liu, Y. Zheng, M. W. Phyu, F. N. Endru, V. Navaneethan, and B. Zhao, "An ultra-low power ecg acquisition and monitoring asic system for wban applications," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 2, pp. 60–70, March 2012.

[40] E. M. E. Hassan and M. Karim, "An fpga-based implementation of a pre-processing stage for ecg signal analysis using dwt," in *2014 Second World Conference on Complex Systems (WCCS)*, pp. 649–654, Nov 2014.

[41] I. Kuon and J. Rose, "Measuring the gap between fpgas and asics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, pp. 203–215, Feb 2007.

[42] M. Seok, S. Hanson, Y.-S. Lin, Z. Foo, D. Kim, Y. Lee, N. Liu, D. Sylvester, and D. Blaauw, "The phoenix processor: A 30pw platform for sensor applications," in *2008 IEEE Symposium on VLSI Circuits*, pp. 188–189, June 2008.

[43] S. R. Sridhara, M. DiRenzo, S. Lingam, S. Lee, R. Blazquez, J. Maxey, S. Ghanem, Y. Lee, R. Abdallah, P. Singh, and M. Goe, "Microwatt embedded processor platform for medical system-on-chip applications," in *2010 Symposium on VLSI Circuits*, pp. 15–16, June 2010.

[44] Y. He, Y. Pu, Z. Ye, S. M. Londono, R. Kleihorst, A. A. Abbo, and H. Corporaal, "Xetalpro: An ultra-low energy and high throughput simd processor," in *Design Automation Conference*, pp. 543–548, June 2010.

[45] Y. Zhang, Y. Shakhsheer, A. T. Barth, H. C. Powell Jr., S. A. Ridenour, M. A. Hanson, J. Lach, and B. H. Calhoun, "Energy efficient design for body sensor nodes," *Journal of Low Power Electronics and Applications*, vol. 1, pp. 109–130, Jan 2011.

[46] A. Y. Dogan, J. Constantin, M. Ruggiero, A. Burg, and D. Atienza, "Multi-core architecture design for ultra-low-power wearable health monitoring systems," in *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 988–993, March 2012.

[47] A. Y. Dogan, R. Braojos, J. Constantin, G. Ansaloni, A. Burg, and D. Atienza, "Synchronizing code execution on ultra-low-power embedded multi-channel signal analysis platforms," in *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 396–399, March 2013.

[48] M.-H. Lee, H. Singh, G. Lu, N. Bagherzadeh, F. J. Kurdahi, E. M. Filho, and V. C. Alves, "Design and implementation of the morphosys reconfigurable computing processor," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 24, pp. 147–164, Mar 2000.

[49] S. Akselrod, D. Gordon, F. Ubel, D. Shannon, A. Berger, and R. Cohen, "Power spectrum analysis of heart rate fluctuation: a quantitative probe of beat-to-beat cardiovascular control," *Science*, vol. 213, pp. 220–222, Jul 1981.

[50] G. Karakonstantis, A. Sankaranarayanan, M. M. Sabry, D. Atienza, and A. Burg, "A quality-scalable and energy-efficient approach for spectral analysis of heart rate variability," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1–6, March 2014.

[51] A. Avci, S. Bosch, M. Marin-Perianu, R. Marin-Perianu, and P. Havinga, "Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey," in *23th International Conference on Architecture of Computing Systems 2010*, pp. 1–10, Feb 2010.

[52] J. Lester, T. Choudhury, and G. Borriello, "A practical approach to recognizing physical activities," in *Proceedings of the 4th International Conference on Pervasive Computing*, PERVASIVE'06, (Berlin, Heidelberg), pp. 1–16, Springer-Verlag, 2006.

[53] F. Casamassima, E. Farella, and L. Benini, "Context aware power management for motion-sensing body area network nodes," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1–6, March 2014.

[54] C. Park, P. H. Chou, Y. Bai, R. Matthews, and A. Hibbs, "An ultra-wearable, wireless, low power ecg monitoring system," in *2006 IEEE Biomedical Circuits and Systems Conference*, pp. 241–244, Nov 2006.

[55] J. Proulx, R. Clifford, S. Sorensen, D.-J. Lee, and J. Archibald, "Development and evaluation of a bluetooth ekg monitoring sensor," in *19th IEEE Symposium on Computer-Based Medical Systems (CBMS'06)*, pp. 507–511, June 2006.

[56] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *2011 38th Annual International Symposium on Computer Architecture (ISCA)*, pp. 365–376, June 2011.

[57] I. Brumar, M. Casas, M. Moreto, M. Valero, and G. S. Sohi, "Atm: Approximate task memoization in the runtime system," in *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 1140–1150, May 2017.

[58] A. Panyala, O. Subasi, M. Halappanavar, A. Kalyanaraman, D. Chavarria-Miranda, and S. Krishnamoorthy, "Approximate computing techniques for iterative graph algorithms," in *2017 IEEE 24th International Conference on High Performance Computing (HiPC)*, pp. 23–32, Dec 2017.

[59] G. C. Calafiore, "Randomized algorithms for robustness analysis: a distributed approach," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pp. 7030–7035, Dec 2009.

[60] A. Lingamneni, C. Enz, J. Nagel, K. Palem, and C. Piguet, "Energy parsimonious circuit design through probabilistic pruning," in *2011 Design, Automation Test in Europe*, pp. 1–6, March 2011.

[61] J. Schlachter, V. Camus, C. Enz, and K. V. Palem, "Automatic generation of inexact digital circuits by gate-level pruning," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 173–176, May 2015.

[62] J. Schlachter, V. Camus, K. V. Palem, and C. Enz, "Design and applications of approximate circuits by gate-level pruning," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, pp. 1694–1702, May 2017.

[63] D. Bortolotti, A. Bartolini, C. Weis, D. Rossi, and L. Benini, "Hybrid memory architecture for voltage scaling in ultra-low power multi-core biomedical processors," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1–6, March 2014.

[64] A. Raha, S. Sutar, H. Jayakumar, and V. Raghunathan, "Quality configurable approximate dram," *IEEE Transactions on Computers*, vol. 66, pp. 1172–1187, July 2017.

[65] F. Frustaci, M. Khayatzadeh, D. Blaauw, D. Sylvester, and M. Alioto, "Sram for error-tolerant applications with dynamic energy-quality management in 28 nm cmos," *IEEE Journal of Solid-State Circuits*, vol. 50, pp. 1310–1323, May 2015.

[66] J. Wang and J. Chong, "Adaptive multi-level block truncation coding for frame memory reduction in lcd overdrive," *IEEE Transactions on Consumer Electronics*, vol. 56, pp. 1130–1136, May 2010.

[67] P. Weckx, B. Kaczer, M. Toledano-Luque, P. Raghavan, J. Franco, P. J. Roussel, G. Groeseneken, and F. Catthoor, "Implications of bti-induced time-dependent statistics on yield estimation of digital circuits," *IEEE Transactions on Electron Devices*, vol. 61, pp. 666–673, March 2014.

[68] ITRS. 2016, "International Technology Roadmap for Semiconductors." www.itrs2.net, 2016.

[69] A. Sánchez-Macián, P. Reviriego, and J. A. Maestro, "Hamming sec-daed and extended hamming sec-ded-taed codes through selective shortening and bit placement," *IEEE Transactions on Device and Materials Reliability*, vol. 14, pp. 574–576, March 2014.

[70] M. A. Breuer, "Multi-media applications and imprecise computation," in *8th Euromicro Conference on Digital System Design (DSD'05)*, pp. 2–7, Aug 2005.

[71] P. Gupta, Y. Agarwal, L. Dolecek, N. Dutt, R. K. Gupta, R. Kumar, S. Mitra, A. Nicolau, T. S. Rosing, M. B. Srivastava, S. Swanson, and D. Sylvester, "Underdesigned and opportunistic computing in presence of hardware variability," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, pp. 8–23, Jan 2013.

[72] P. K. Krause and I. Polian, "Adaptive voltage over-scaling for resilient applications," in *2011 Design, Automation Test in Europe*, pp. 1–6, March 2011.

[73] J. W. S. Liu, K. . Lin, W. . Shih, A. C. . Yu, J. . Chung, and W. Zhao, "Algorithms for scheduling imprecise computations," *Computer*, vol. 24, pp. 58–68, May 1991.

[74] G. V. Varatkar and N. R. Shanbhag, "Error-resilient motion estimation architecture," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, pp. 1399–1412, Oct 2008.

[75] M. Cannizzaro, S. Beer, J. Cortadella, R. Ginosar, and L. Lavagno, "Saferazor: Metastability-robust adaptive clocking in resilient circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, pp. 2238–2247, Sept 2015.

[76] X. Li and D. Yeung, "Application-level correctness and its impact on fault tolerance," in *2007 IEEE 13th International Symposium on High Performance Computer Architecture*, pp. 181–192, Feb 2007.

[77] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, pp. 147–160, April 1950.

[78] F. Massé, M. V. Bussel, A. Serteyn, J. Arends, and J. Penders, "Miniaturized wireless ecg monitor for real-time detection of epileptic seizures," *ACM Trans. Embed. Comput. Syst.*, vol. 12, pp. 102:1–102:21, July 2013.

[79] J. Milosevic, A. Dittrich, A. Ferrante, M. Malek, C. R. Quiros, R. Braojos, G. Ansaloni, and D. Atienza, "Risk assessment of atrial fibrillation: A failure prediction approach," in *Computing in Cardiology 2014*, pp. 801–804, Sept 2014.

[80] L. Sörnmo and P. Laguna, *Bioelectrical signal processing in cardiac and neurological applications*. (Academic Press, Burlington, USA, 2005.

[81] C. Chou, S. Tseng, E. Chua, Y. Lee, W. Fang, and H. Huang, "Advanced ecg processor with hrv analysis for real-time portable health monitoring," in *2011 IEEE International Conference on Consumer Electronics -Berlin (ICCE-Berlin)*, pp. 172–175, Sept 2011.

[82] G. Karakonstantis, A. Sankaranarayanan, and A. Burg, "Low complexity spectral analysis of heart-rate-variability through a wavelet based fft," in *2012 Computing in Cardiology*, pp. 285–288, Sept 2012.

[83] W. H. Press and G. B. Rybicki, "Fast algorithm for spectral analysis of unevenly sampled data," *Astrophysical Journal - ASTROPHYS J*, vol. 338, pp. 277–280, 03 1989.

[84] N. Boichat, N. Khaled, F. Rincon, and D. Atienza, "Wavelet-based ecg delineation on a wearable embedded sensor platform," in *2009 Sixth International Workshop on Wearable and Implantable Body Sensor Networks*, pp. 256–261, June 2009.

[85] "The PAF Prediction Challenge Database." https://physionet.org/physiobank/database/afpdb/, March 2001.

[86] Muralimanohar, N., Balasubramonian, R., Jouppi, N.P., "'CACTI 6.0: A tool to model large caches," tech. rep., 2009.

[87] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 469–480, Dec 2009.

[88] ARM Holdings, "Cortex M4." https://developer.arm.com/products/processors/cortex-m/cortex-m4, 2018.

[89] S. J. E. Wilton and N. P. Jouppi, "Cacti: an enhanced cache access and cycle time model," *IEEE Journal of Solid-State Circuits*, vol. 31, pp. 677–688, May 1996.

[90] ARM Holdings, "Cortex M3." https://developer.arm.com/products/processors/cortex-m/cortex-m3, 2018.

[91] R. Winchell and D. Hoyt, "Spectral analysis of heart rate variability in the icu: a measure of autonomic function," *J. Surg. Res.*, vol. 63, no. 1, p. 11–16, 1996.

[92] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surv.*, vol. 48, pp. 62:1–62:33, Mar. 2016.

[93] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *Proceedings 2003. Design Automation Conference (IEEE Cat. No.03CH37451)*, pp. 338–342, June 2003.

[94] S. Khare and S. Jain, "Prospects of near-threshold voltage design for green computing," in *2013 26th International Conference on VLSI Design and 2013 12th International Conference on Embedded Systems*, pp. 120–124, Jan 2013.

[95] S. Mukhopadhyay, H. Mahmoodi, and K. Roy, "Modeling of failure probability and statistical design of sram array for yield enhancement in nanoscaled cmos," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, pp. 1859–1880, Dec 2005.

[96] U. R. Karpuzcu, K. B. Kolluru, N. S. Kim, and J. Torrellas, "Varius-ntv: A microarchitectural model to capture the increased sensitivity of manycores to process variations at near-threshold voltages," in *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)*, pp. 1–11, June 2012.

[97] P. Ghanta, S. Vrudhula, R. Panda, and J. Wang, "Stochastic power grid analysis considering process variations," in *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 2*, DATE '05, (Washington, DC, USA), pp. 964–969, IEEE Computer Society, 2005.

[98] X. Zhang, T. Tong, S. Kanev, S. K. Lee, G. Wei, and D. Brooks, "Characterizing and evaluating voltage noise in multi-core near-threshold processors," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 82–87, Sept 2013.

[99] S. S. Mukherjee, J. Emer, and S. K. Reinhardt, "The soft error problem: an architectural perspective," in *11th International Symposium on High-Performance Computer Architecture*, pp. 243–247, Feb 2005.

[100] S. Basu, L. Duch, R. Braojos, G. Ansaloni, L. Pozzi, and D. Atienza, "An inexact ultra-low power bio-signal processing architecture with lightweight error recovery," *ACM Trans. Embed. Comput. Syst.*, vol. 16, pp. 159:1–159:19, Sept. 2017.

[101] J. Constantin, A. Dogan, O. Andersson, P. Meinerzhagen, J. N. Rodrigues, D. Atienza, and A. Burg, "Tamarisc-cs: An ultra-low-power application-specific processor for compressed sensing," in *2012 IEEE/IFIP 20th International Conference on VLSI and System-on-Chip (VLSI-SoC)*, pp. 159–164, Oct 2012.

[102] "MIT-BIH Normal Sinus Rhythm Database." https://www.physionet.org/physiobank/database/nsrdb/, February 2012.

[103] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, pp. 1289–1306, April 2006.

[104] Synopsys, "ASIP Designer." www.synopsys.com/dw/ipdir.php?ds=asip-designer, 2017.

[105] United Microelectronics Corporation, "65 Nanometer." http://www.umc.com/English/pdf/UMC%2065nm.pdf, 2005.

[106] D. Evans, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything." http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf, April 2011.

[107] M. Lazarescu and L. Lavagno, *Wireless Sensor Networks*. Dordrecht: (Springer Science+Business Media, 2017.

[108] A. Pantelopoulos and N. G. Bourbakis, "A survey on wearable sensor-based systems for health monitoring and prognosis," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, pp. 1–12, Jan 2010.

[109] R. Fensli, E. Gunnarson, and T. Gundersen, "A wearable ecg-recording system for continuous arrhythmia monitoring in a wireless tele-home-care situation," in *18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05)*, pp. 407–412, June 2005.

[110] N. Oliver and F. Flores-Mangas, "Healthgear: a real-time wearable system for monitoring and analyzing physiological signals," in *International Workshop on Wearable and Implantable Body Sensor Networks (BSN'06)*, pp. 4 pp.–64, April 2006.

[111] C. M. M. Sung and A. Pentland, "Wearable feedback systems for rehabilitation," *Journal of NeuroEngineering and Rehabilitation*, vol. 2, pp. 2–17, June 2005.

[112] J. Kwong and A. P. Chandrakasan, "An energy-efficient biomedical signal processing platform," *IEEE Journal of Solid-State Circuits*, vol. 46, pp. 1742–1753, July 2011.

[113] S.-C. Huang, H.-M. Wang, and W.-Y. Chen, "A ±6ms-accuracy, 0.68mm2, and 2.21 μw qrs detection asic," *VLSI Des.*, vol. 2012, pp. 20:20–20:20, Jan. 2012.

[114] M. W. Phyu, Y. Zheng, B. Zhao, L. Xin, and Y. S. Wang, "A real-time ecg qrs detection asic based on wavelet multiscale analysis," in *2009 IEEE Asian Solid-State Circuits Conference*, pp. 293–296, Nov 2009.

[115] F. Conti, A. Marongiu, and L. Benini, "Synthesis-friendly techniques for tightly-coupled integration of hardware accelerators into shared-memory multi-core clusters," in *2013 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pp. 1–10, Sept 2013.

[116] Texas Instruments , "Texas instruments MSP430 family." http://www.ti.com/lsds/ti/microcontrollers_16-bit_32-bit/msp/overview.page?DCMP=MCU_%2520other&HQS=msp430, 2016.

[117] M. Johnson, M. Healy, P. van de Ven, M. J. Hayes, J. Nelson, T. Newe, and E. Lewis, "A comparative review of wireless sensor network mote technologies," in *SENSORS, 2009 IEEE*, pp. 1439–1442, Oct 2009.

[118] "Shimmer Sensing." http://www.shimmersensing.com/.Las, 2016.

[119] A. Burns, B. R. Greene, M. J. McGrath, T. J. O'Shea, B. Kuris, S. M. Ayer, F. Stroiescu, and V. Cionca, "Shimmer™ – a wireless sensor platform for noninvasive biomedical research," *IEEE Sensors Journal*, vol. 10, pp. 1527–1534, Sept 2010.

[120] R. Fasthuber, F. Catthoor, P. Raghavan, and F. Naessens, *Energy-efficient communication processors: Design and implementation for emerging wireless systems*. 03 2014.

[121] I. A. Khatib, A. Jantsch, and D. Bertozzi, "Mpsoc ecg biochip: a multiprocessor system-on-chip for real-time human heart monitoring and analysis," in *In Proceedings of Computing Frontiers 2006 (CF'06), Iscia*, pp. 21–28, 2006.

[122] H. Kim, S. Kim, N. V. Helleputte, A. Artes, M. Konijnenburg, J. Huisken, C. V. Hoof, and R. F. Yazicioglu, "A configurable and low-power mixed signal soc for portable ecg monitoring applications," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, pp. 257–267, April 2014.

[123] F. Bouwens, M. Berekovic, A. Kanstein, and G. Gaydadjiev, "Architectural exploration of the adres coarse-grained reconfigurable array," in *Reconfigurable Computing: Architectures, Tools and Applications* (P. C. Diniz, E. Marques, K. Bertels, M. M. Fernandes, and J. M. P. Cardoso, eds.), (Berlin, Heidelberg), pp. 1–13, Springer Berlin Heidelberg, 2007.

[124] N. Ozaki, Y. Yasuda, M. Izawa, Y. Saito, D. Ikebuchi, H. Amano, H. Nakamura, K. Usami, M. Namiki, and M. Kondo, "Cool mega-arrays: Ultralow-power reconfigurable accelerator chips," *IEEE Micro*, vol. 31, pp. 6–18, Nov 2011.

[125] K. Patel and C. J. Bleakley, "Coarse grained reconfigurable array based architecture for low power real-time seizure detection," *Journal of Signal Processing Systems*, vol. 82, pp. 55–68, Jan 2016.

[126] G. Ansaloni, L. Pozzi, K. Tanimura, and N. Dutt, "Slack-aware scheduling on coarse grained reconfigurable arrays," in *2011 Design, Automation Test in Europe*, pp. 1–4, March 2011.

[127] P. Theocharis and B. D. Sutter, "A bimodal scheduler for coarse-grained reconfigurable arrays," *ACM Trans. Archit. Code Optim.*, vol. 13, pp. 15:1–15:26, June 2016.

[128] H. Mitani, K. Matsubara, H. Yoshida, T. Hashimoto, H. Yamakoshi, S. Abe, T. Kono, Y. Taito, T. Ito, T. Krafuji, K. Noguchi, H. Hidaka, and T. Yamauchi, "7.6 a 90nm embedded 1t-monos flash macro for automotive applications with 0.07mj/8kb rewrite energy and endurance over 100m cycles under tj of 175°c," in *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 140–141, Jan 2016.

[129] Y. Taito, M. Nakano, H. Okimoto, D. Okada, T. Ito, T. Kono, K. Noguchi, H. Hidaka, and T. Yamauchi, "7.3 a 28nm embedded sg-monos flash macro for automotive achieving 200mhz read operation and 2.0mb/s write throughput at ti, of 170°c," in *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, pp. 1–3, Feb 2015.

[130] X. Dong, X. Wu, G. Sun, Y. Xie, H. Li, and Y. Chen, "Circuit and microarchitecture evaluation of 3d stacking magnetic ram (mram) as a universal memory replacement,"

in *Proceedings of the 45th Annual Design Automation Conference*, DAC '08, (New York, NY, USA), pp. 554–559, ACM, 2008.

[131] A. Nigam, C. W. Smullen, V. Mohan, E. Chen, S. Gurumurthi, and M. R. Stan, "Delivering on the promise of universal memory for spin-transfer torque ram (stt-ram)," in *IEEE/ACM International Symposium on Low Power Electronics and Design*, pp. 121–126, Aug 2011.

[132] F. Sampaio, M. Shafique, B. Zatt, S. Bampi, and J. Henkel, "Energy-efficient architecture for advanced video memory," in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 132–139, Nov 2014.

[133] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan, "Relaxing non-volatility for fast and energy-efficient stt-ram caches," in *2011 IEEE 17th International Symposium on High Performance Computer Architecture*, pp. 50–61, Feb 2011.

[134] Z. Sun, X. Bi, H. Li, W. Wong, Z. Ong, X. Zhu, and W. Wu, "Multi retention level stt-ram cache designs with a dynamic refresh scheme," in *2011 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 329–338, Dec 2011.

[135] W. Xu, H. Sun, X. Wang, Y. Chen, and T. Zhang, "Design of last-level on-chip cache using spin-torque transfer ram (stt ram)," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, pp. 483–493, March 2011.

[136] B. Ransford, J. Sorber, and K. Fu, "Mementos: System support for long-running computation on rfid-scale devices," in *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVI, (New York, NY, USA), pp. 159–170, ACM, 2011.

[137] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A novel architecture of the 3d stacked mram l2 cache for cmps," in *2009 IEEE 15th International Symposium on High Performance Computer Architecture*, pp. 239–249, Feb 2009.

[138] H. Wei, M. Shulaker, H. . P. Wong, and S. Mitra, "Monolithic three-dimensional integration of carbon nanotube fet complementary logic circuits," in *2013 IEEE International Electron Devices Meeting*, pp. 19.7.1–19.7.4, Dec 2013.

[139] M. M. Shulaker, T. F. Wu, A. Pal, L. Zhao, Y. Nishi, K. Saraswat, H. . P. Wong, and S. Mitra, "Monolithic 3d integration of logic and memory: Carbon nanotube fets, resistive ram, and silicon fets," in *2014 IEEE International Electron Devices Meeting*, pp. 27.4.1–27.4.4, Dec 2014.

[140] M. M. Shulaker, T. F. Wu, M. M. Sabry, H. Wei, H. . P. Wong, and S. Mitra, "Monolithic 3d integration: A path from concept to reality," in *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1197–1202, March 2015.

[141] J. Yang-Scharlotta, M. Fazio, M. Amrbar, M. White, and D. Sheldon, "Reliability characterization of a commercial taox-based reram," in *2014 IEEE International Integrated Reliability Workshop Final Report (IIRW)*, pp. 131–134, Oct 2014.

[142] I. Micron Technology, "Wear-Leveling Techniques in NAND Flash Devices," tech. rep., 2008.

[143] G. Ansaloni, P. Bonzini, and L. Pozzi, "Egra: A coarse grained reconfigurable architectural template," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, pp. 1062–1074, June 2011.

[144] B. Mei, S. Vernalde, D. Verkest, H. D. Man, and R. Lauwereins, "Exploiting loop-level parallelism on coarse-grained reconfigurable architectures using modulo scheduling," *IEE Proceedings - Computers and Digital Techniques*, vol. 150, pp. 255–, Sept 2003.

[145] L. Chen and T. Mitra, "Shared reconfigurable fabric for multi-core customization," in *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 830–835, June 2011.

[146] C. Lattner and V. Adve, "Llvm: a compilation framework for lifelong program analysis transformation," in *International Symposium on Code Generation and Optimization, 2004. CGO 2004.*, pp. 75–86, March 2004.

[147] H. B. Ramakrishna Rau, "Iterative modulo scheduling." http://www.hpl.hp.com/techreports/94/HPL-94-115.pdf, 1995.

[148] Mentor Graphics, ModelSim Software. https://www.mentor.com/products/fv/modelsim/, Nov. 2017.

[149] Synopsys, Design Compiler Software. https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/design-compiler-graphical.html, Nov. 2017.

[150] L. Duch, S. Basu, R. Braojos, D. Atienza, G. Ansaloni, and L. Pozzi, "A multi-core reconfigurable architecture for ultra-low power bio-signal analysis," in *2016 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 416–419, Oct 2016.

[151] S. Yang and D. Bian, "Automatic detection of qrs onset in ecg signals," in *2008 IEEE International Symposium on IT in Medicine and Education*, pp. 291–294, Dec 2008.

[152] R. Braojos, G. Ansaloni, and D. Atienza, "A methodology for embedded classification of heartbeats using random projections," in *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 899–904, March 2013.

[153] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits," *Proceedings of the IEEE*, vol. 98, pp. 253–266, Feb 2010.

[154] A. Y. Dogan, R. Braojos, J. Constantin, G. Ansaloni, A. Burg, and D. Atienza, "Synchronizing code execution on ultra-low-power embedded multi-channel signal analysis platforms," in *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 396–399, March 2013.

[155] J. L. Hennessy and D. A. Patterson, *Computer Architecture; A Quantitative Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1st ed., 1992.

[156] "PhysioBank." http://www.physionet.org/physiobank/, Aug. 2016.

[157] Y. Zigel, A. Cohen, and A. Katz, "The weighted diagnostic distortion (wdd) measure for ecg signal compression," vol. 47, pp. 1424–30, 12 2000.

[158] R. Strenz, "Embedded flash technologies and their applications: Status amp; outlook," in *2011 International Electron Devices Meeting*, pp. 9.4.1–9.4.4, Dec 2011.

[159] H. Hidaka, "Evolution of embedded flash memory technology for mcu," in *2011 IEEE International Conference on IC Design Technology*, pp. 1–4, May 2011.

[160] Infineon Technologies, ""Embedded Flash Technologies: Enabler for Automotive μCs & Smartcards." Workshop on Innovative Memory Technologies, MINATEC, Grenoble, 2012.

[161] E. Vianello, O. Thomas, M. Harrand, S. Onkaraiah, T. Cabout, B. Traoré, T. Diokh, H. Oucheikh, L. Perniola, G. Molas, P. Blaise, J. F. Nodin, E. Jalaguier, and B. D. Salvo, "Back-end 3d integration of hfo2-based rrams for low-voltage advanced ic digital design," in *Proceedings of 2013 International Conference on IC Design Technology (ICICDT)*, pp. 235–238, May 2013.

[162] H. Yu, K. Lin, Y. Chih, and J. Chang, "A 40nm split gate embedded flash macro with flexible 2-in-1 architecture, code memory with 140mhz read speed and data memory with 1m cycles endurance," in *2017 Symposium on VLSI Circuits*, pp. C198–C199, June 2017.

[163] Panasonic Corporation, "ReRAM Embedded Super Low-Power Consumption MCU MN101L." https://industrial.panasonic.com/ww/products/semiconductors/microcomputers/mn101l.

[164] W. C. Shen, C. Y. Mei, Y. D. Chih, S. Sheu, M. Tsai, Y. King, and C. J. Lin, "High-K Metal Gate Contact RRAM (CRRAM) in Pure 28nm CMOS Logic Process," in *International Electron Devices Meeting*, pp. 31.6.1–31.6.4, Dec. 2012.

[165] H. P. Wong, H. Lee, S. Yu, Y. Chen, Y. Wu, P. Chen, B. Lee, F. T. Chen, and M. Tsai, "Metal–oxide rram," *Proceedings of the IEEE*, vol. 100, pp. 1951–1970, June 2012.

[166] R. Fackenthal, M. Kitagawa, W. Otsuka, K. Prall, D. Mills, K. Tsutsui, J. Javanifard, K. Tedrow, T. Tsushima, Y. Shibahara, and G. Hush, "19.7 a 16gb reram with 200mb/s

write and 1gb/s read in 27nm technology," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 338–339, Feb 2014.

[167] M. E. Sinangil, H. Mair, and A. P. Chandrakasan, "A 28nm High-Density 6T SRAM with Optimized Peripheral-Assist Circuits for Operation down to 0.6V," in *IEEE International Solid-State Circuits Conference*, pp. 260–262, Feb. 2011.

[168] G. Sassine, C. Nail, L. Tillie, D. A. Robayo, A. Levisse, C. Cagli, K. E. Hajjam, J. Nodin, E. Vianello, M. Bernard, G. Molas, and E. Nowak, "Sub-pJ Consumption and Short Latency Time in RRAM Arrays for High Endurance Applications," in *IEEE International Reliability Physics Symposium (IRPS)*, Mar. 2018.

[169] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, pp. 994–1007, July 2012.

[170] S. J. E. Wilton and N. P. Jouppi, "Cacti: an enhanced cache access and cycle time model," *IEEE Journal of Solid-State Circuits*, vol. 31, pp. 677–688, May 1996.

[171] C. Nail, G. Molas, P. Blaise, G. Piccolboni, B. Sklenard, C. Cagli, M. Bernard, A. Roule, M. Azzaz, E. Vianello, C. Carabasse, R. Berthier, D. Cooper, C. Pelissier, T. Magis, G. Ghibaudo, C. Vallèe, D. Bedeau, O. Mosendz, B. D. Salvo, and L. Perniola, "Understanding RRAM Endurance, Retention and Window Margin Trade-Off Using Experimental Results and Simulations," in *IEEE International Electron Devices Meeting (IEDM)*, Dec. 2016.

[172] S. Schechter, G. H. Loh, K. Strauss, and D. Burger, "Use ECP, Not ECC, for Hard Failures in Resistive Memories," in *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA)*, pp. 141–152, ACM, 2010.

[173] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali, "Enhancing Lifetime and Security of PCM-Based Main Memory with Start-Gap Wear Leveling," in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec. 2009.

[174] S. Basu, P. G. d. Valle, G. Karakonstantis, G. Ansaloni, and D. Atienza, "Heterogeneous error-resilient scheme for spectral analysis in ultra-low power wearable electrocardiogram devices," in *2015 IEEE Computer Society Annual Symposium on VLSI*, pp. 268–273, July 2015.

[175] S. Basu, P. G. D. Valle, G. Karakonstantis, G. Ansaloni, L. Pozzi, and D. Atienza, "Inexact-aware architecture design for ultra-low power bio-signal analysis," *IET Computers Digital Techniques*, vol. 10, no. 6, pp. 306–314, 2016.

[176] L. Duch, S. Basu, M. Peón-Quirós, G. Ansaloni, L. Pozzi, and D. Atienza, "i-dps cgra: An interleaved-datapaths reconfigurable accelerator for embedded bio-signal processing," *IEEE Embedded Systems Letters*, pp. 1–1, 2018.

[177] S. Basu, L. Duch, M. Peón-Quirós, D. Atienza, G. Ansaloni, and L. Pozzi, "Heterogeneous and inexact: Maximizing power efficiency of edge computing sensors for health monitoring applications," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2018.

[178] C. Lattner and V. Adve, "Llvm: a compilation framework for lifelong program analysis amp; transformation," in *International Symposium on Code Generation and Optimization, 2004. CGO 2004.*, pp. 75–86, March 2004.

[179] G. Zacharopoulos and L. Pozzi, "Clrfreqcfgprinter: A tool for frequency annotated control flow graphs generation," tech. rep., 03 2017.

[180] T. Peyret, G. Corre, M. Thevenin, K. Martin, and P. Coussy, "An automated design approach to map applications on cgras," in *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*, GLSVLSI '14, (New York, NY, USA), pp. 229–230, ACM, 2014.

[181] H. Park, K. Fan, S. Mahlke, T. Oh, H. Kim, and H. Kim, "Edge-centric modulo scheduling for coarse-grained reconfigurable architectures," in *2008 International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pp. 166–176, Oct 2008.

Soumya Basu

Final year PhD student, EPFL, Switzerland

Personal details

Nationality	Indian
Marital status	Single
Permit	Swiss B work permit
Date of birth	20^{th} December, 1989

Skills

- Expert in embedded computer programming (Hardware & Software), with 4+ years of experience in academia and industry: C, VHDL, C++, Verilog.
- Proficient user of embedded systems design platforms: FPGA, micro-controllers, tools from Xilinx, Synopsys, Cadence, Vector, Matlab.

Education

2014-2019 **PhD**, *École Polytechnique Fédérale de Lausanne (EPFL)*, Lausanne, *Switzerland. Degree expected in **January 2019***.
Embedded Systems Laboratory, ESL-EPFL. Advisor: *Prof. David Atienza Alonso*

2011–2013 **Master of Science**, *Politecnico di Torino*, Turin, *Italy*.
Electronic Engineering, specializing in Embedded Systems.

Work Experience

05/18–07/18 **PhD Internship**, *Stanford University, Robust Systems Group*, Stanford, USA.
Development of nano-technology enabled reliable smart IoT sensors, based on emerging non-volatile ReRAM technologies.

11/13–05/14 **Post-MS Internship**, *Thales Alenia Space SpA.*, Turin, Italy.
CAN bus solutions for modern spacecraft data handling systems: Study, development, and testing of CAN bus architectures for telecommunication and scientific spacecraft platforms at hardware and software level.

06/10–08/10 **Scientific Associate**, *European Organization for Nuclear Research (CERN)*, Geneva, Switzerland.
Design, simulation and test of a circuit for the generation of a digital pulse generator based on an FPGA.

Route Cantonale 35 – 1025 St-Sulpice (VD) – Switzerland
+41 768 199 624 • *+41 21 69 311 41*
soumya.basu9@gmail.com

Languages

- English — *Full proficiency, C2*
- French, Italian, Hindi — *Professional user, B2*
- German, Spanish — *Basic user, A1*
- Bengali — *Mother tongue*

Computer skills

Programming languages C, C++, SystemC, C#, JAVA, CAPL, scripting.

Designer tools: Circuit designing tools from Synopsys, Xilinx, Cadence.

Hardware tools VHDL, FPGA, CGRA, Verilog, Micro-controller programming.

Operating systems: Unix, Mac, Windows, Jennic JenOS RTOS.

Softwares Git, Matlab, Simulink, CANalyzer, CANoe, Eclipse.

Text editors: MS Office tools, Latex.

Doctoral thesis

Title *Hardware/Software Co-design Methodologies for Ultra-low Power Bio-Signal Processing Architectures*

Description Design and optimization of ultra-low power computing architectures for autonomous bio-signal analysis and processing, targeted for wearable health monitoring devices.

Skills acquired Strong skills in VHDL, C, C++, SystemC and Matlab, with experience in Bluetooth LE and Zigbee, working with them over 4 years during PhD.

Master thesis

Title *The CAN Bus In Space: Improvement of The European Space Agency's HurriCANe IP-Core*

Description Modify and update the HurriCANe 5.2.4 IP-Core of the European Space Agency to meet the functional and performance requirements that are stated by the ESA.

Skills acquired Strong skills in Vector CANalyzer, CANoe, VHDL, C, CAPL coding and Synopsys tools, working for over a year in industry and in academia.

Honors and awards

- Awarded EPFL-Stanford Exchange Fellowship, 2018.
- Selected for Summer Internship at CERN, Geneva 2010.
- Selected in the Swissnex India Academy-Industry Training camp aimed at future entrepreneurs, Bangalore 2016.
- President, YUVA Indian students' association of EPFL/UNIL, 2015-2016.
- Represented and captained the Swiss university team at the European University Games (Table Tennis), Zagreb 2016.
- Swiss University champion, in Table Tennis (singles & team), 2015.

Route Cantonale 35 – 1025 St-Sulpice (VD) – Switzerland
+41 768 199 624 • +41 21 69 311 41
soumya.basu9@gmail.com

References

- Prof. David Atienza Alonso, *director of ESL-EPFL:* david.atienza@epfl.ch
- Prof. Laura Pozzi, *professor, faculty of informatics, USI Lugano:* laura.pozzi@usi.ch
- Dr. Giovanni Ansaloni, *post-doc researcher, faculty of informatics, USI Lugano:* giovanni.ansaloni@usi.ch
- Dr. Miguel Peon Quiros, *post-doc researcher, ESL-EPFL:* miguel.peon@epfl.ch
- Dr. Ruben Braojos Lopez, *senior software engineer, SmartCardia SA, Lausanne:* ruben.braojos@smartcardia.com

Publications

[1] S. Basu, L. Duch, M. Peon Quiros, G. Ansaloni, L. Pozzi and D. Atienza **Heterogeneous and Inexact: Maximizing Power Efficiency of Edge Computing Sensors for Health Monitoring Applications** in *The International Symposium on Circuits and Systems (ISCAS)*, 2018

[2] L. Duch, S. Basu, M. Peon Quiros, G. Ansaloni, L. Pozzi and D. Atienza: **i-DPs CGRA: An Interleaved-Datapaths Reconfigurable Accelerator for Embedded Bio-signal Processing** in *IEEE Embedded Systems Letters (ESL)*, 2018

[3] S. Basu, L. Duch, R. Braojos, G. Ansaloni, L. Pozzi and D. Atienza **An Inexact Ultra-low Power Bio-signal Processing Architecture With Lightweight Error Recovery** in *TACM Transactions on Embedded Computing Systems*, 2017

[4] L. Duch, S. Basu, R. Braojos, G. Ansaloni, L. Pozzi and D. Atienza: **HEAL-WEAR: an Ultra-Low Power Heterogeneous System for Bio-Signal Analysis** in *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2017

[5] S. Basu, P. Garcia del Valle, G. Karakonstantis, G. Ansaloni, L. Pozzi and D. Atienza **Inexact-Aware Architecture Design for Ultra-Low Power Bio-Signal Analysis** in *IET Computers & Digital Techniques*, 2016

[6] L. Duch, S. Basu, R. Braojos, G. Ansaloni, L. Pozzi and D. Atienza: **A Multi-Core Reconfigurable Architecture for Ultra-Low Power Bio-Signal Analysis** in *Biomedical Circuits and Systems (BioCAS)*, 2016

[7] S. Basu, P. Garcia del Valle, G. Ansaloni, G. Karakonstantis and D. Atienza **Heterogeneous Error-Resilient Scheme for Spectral Analysis in Ultra-Low Power Wearable Electrocardiogram Devices** in *IEEE Annual Symposium on VLSI (ISVLSI)*, 2015

Route Cantonale 35 – 1025 St-Sulpice (VD) – Switzerland
+41 768 199 624 • +41 21 69 311 41
soumya.basu9@gmail.com

**Ce document a été imprimé au Centre d'impression EPFL,
imprimerie climatiquement neutre, certifiée myClimate.**