Master's Thesis

Design and Implementation of a Dataflow-Oriented Hardware-Accelerated Processing Architecture for Ultra-Low-Power Sensor Nodes

Daniel Wittibschlager

Institute for Technical Informatics
Graz University of Technology

Interuniversity Microelectronics Center NL
Holst Centre
High Tech Campus 31
5656 Eindhoven, The Netherlands

Reviewer: Ass.-Prof. Dipl.-Ing. Dr. techn. Christian Steger

Advisor: Ass.-Prof. Dipl.-Ing. Dr. techn. Christian Steger
Dipl.-Ing. Dr. techn. Christian Bachmann (IMEC)

Eindhoven/Graz

January, 2013

This page is intentionally left blank.

Abstract

Using wireless sensor networks for medical application requires a high user acceptance, which necessitates small device form factors. Batteries required for power supply contribute significantly to the size of on-body wireless sensor nodes. Therefore reducing the power consumption of those on-body nodes is the uttermost crucial issue to address. Processing units are major contributors to a sensor node's power consumption. An essential challenge in processing unit design for sensor platforms is an implementation allowing low-power consumption, while providing the potential for executing the computation-extensive algorithms. This thesis presents an exploration of a dataflow-oriented communication concept for a hardware-accelerated processing architecture. Deploying multiple accelerators to support a general purpose processor (GPP) allows for combining high performance, low power consumption and flexibility. A reconfigurable unidirectional data stream enables power-efficient and flexible communication between the deployed processing engines. Furthermore, it provides control signals for efficient power management. The system concept derived in this thesis is applied to a low-power processing system for Electrocardiography monitoring applications. The resulting architecture is synthesized as well as placed and routed for a 180 nm process in order to derive accurate power consumption estimations. Power consumption estimates and area results are presented and compared to the baseline implementation.

Keywords: Hardware-accelerated Processing Architecture, Low-Power, Dataflow-oriented Processing, Streaming-based Communication, Electrocardiography, R-peak detection

Kurzfassung

Damit sich drahtlose Sensornetzwerke in medizinischen Bereichen etablieren können, ist es notwendig die Geräte so klein wie möglich zu gestalten um Akzeptanz bei den Patienten zu erreichen. Die Batterie trägt maßgeblich zu der Größe von den am Körper angebrachten Sensoren bei, darum ist die Reduzierung des Leistungsverbrauchs dieser Systeme ein Problem, das einer Lösung bedarf. Die Verarbeitungseinheiten tragen einen Großteil zum Leistungsbedarf der Sensoren bei. Eine essentielle Herausforderung während dem Design der Verarbeitungseinheit ist es, eine Implementierung zu schaffen, die einerseits ein Minimum an Leistung benötigt und andererseits das Potential besitzt, rechenintensive Algorithmen zu berechnen. In dieser Arbeit wird ein datenflussorientiertes Kommunikationskonzept für hardwarebeschleunigte Verarbeitungsarchitekturen implementiert. Die Verwendung von verschiedenen Beschleunigern, welche eine zentrale Recheneinheit unterstützen, kombiniert niedrigen Leistungsbedarf und Flexibilität mit hohen Performanceansprüchen. Ein rekonfigurierbarer, unidirektionaler Datenstrom erlaubt eine leistungseffiziente und flexible Kommunikation zwischen den verschiedenen Rechnereinheiten. Des Weiteren stellt sie Informationen für ein effizientes Leistungsmanagement bereit. Das erstellte Konzept wird an einem Verarbeitungssystem für Elektrokardiogramm-Überwachung erprobt. Die resultierende Architektur wird für einen 180 nm Prozess synthetisiert, um genaue Abschätzungen der Leistungsaufnahme ableiten zu können. Leinsungsabschätzungen sowie physikalischen Charakteristika werden präsentiert.

Stichwörter: Hardwarebeschleunigte Verarbeitungsarchitekturen, Low-Power, datenflussorientierte Verarbeitung, Streaming-basierende Kommunikation, Elektrokardiogramm, R-peak dedektierung

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....
date (signature)

Acknowledgments

This master thesis is the result of a collaboration with the Interuniversity Microelectronics Center Netherlands (IMEC NL) located in Eindhoven and Graz University of Technology. I would like to express my gratitude to all people from both organizations who supported me and made this project possible. Although it is not possible to name everyone, I would like to mention a few.

Special thanks goes to my university adviser, Christian Steger, and to Christian Bachmann at IMEC NL. They made this collaboration possible and offered a great amount of guidance and advice to my thesis.

At IMEC I would like to thank several people: Hyejung Kim for providing the original ECGSoC design - the basis of my thesis - and for all her support in design bring-up. Torfinn Berset for explaining the fundamentals of ECG signal processing. Jos Hulzink who taught me valuable lessons in processor architectures, and helped me to decrypt the compile error messages of the processor design tool chain. Jan Struijt who let me use his memory and always helped with tips and tricks. Mario Konijnenburg and Benjamin Büsze provided the knowledge to synthesize and place & route my design. Pepjin Boer for the discussions on effective algorithm implementation on the designed platform. Jos Huisken and Maryam Ashouei for providing helpful feedback and guidance.

I also would like to thank my colleague Sohan, for the valuable discussions we had at lunch, although he worked on this own thesis at that time.

Concluding, I want to express my gratefulness to my parents for the support they have given me, and to my girlfriend for the patience and moral backup.

Graz, January 2013 Daniel Wittibschlager

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Wireless Sensor Networks for Biomedical Signal Monitoring . . . 2
- 1.3 Thesis Goals . . . 3
- 1.4 Thesis Organization . . . 3

2 Fundamentals of Sensor Nodes and Medical Applications **5**
- 2.1 Introduction . . . 5
- 2.2 Wireless Sensor Networks in Our Society . . . 5
- 2.3 Medical Applications for Wireless Sensor Nodes . . . 6
- 2.4 Electrocardiography . . . 7
  - 2.4.1 Motion Artifact Removal . . . 8
  - 2.4.2 Automatic ECG Analysis . . . 11
- 2.5 Typical Wireless Sensor Node Architecture . . . 12
- 2.6 Sources of Power Consumption and Optimization Techniques . . . 13
  - 2.6.1 Dynamic Power Consumption . . . 14
  - 2.6.2 Static Power Consumption . . . 14
  - 2.6.3 Low Power Techniques . . . 15

3 Processing Systems in Wireless Sensor Nodes **19**
- 3.1 Introduction . . . 19
- 3.2 Processor Architectures in Wireless Sensor Nodes . . . 19
  - 3.2.1 Processor Architectures . . . 20
  - 3.2.2 Overview of Recent WSN Processors . . . 24
  - 3.2.3 Power Management in Ultra-Low-Power Processors . . . 25
- 3.3 On-Chip Communication Architectures in Wireless Sensor Nodes 27
  - 3.3.1 Communication Architectures . . . 27
  - 3.3.2 State-of-the-Art Communication Systems . . . 31
  - 3.3.3 Power Consumption Through Communication . . . 33
  - 3.3.4 Communication Architecture Requirements for ARAs . . . 33
- 3.4 A Low-Power WSN Processing System: ECGSoC . . . 34
  - 3.4.1 Implemented Algorithms . . . 34
  - 3.4.2 Architecture Overview . . . 35
  - 3.4.3 Application-Specific Instruction-Set Processor . . . 36
  - 3.4.4 Application-Specific Accelerators . . . 37
  - 3.4.5 Communication Architecture . . . 37

3.5 Previous Work on ECG-Monitoring Systems . . . 38
3.5.1 ASIC Implementations . . . 38
3.5.2 Configurable Implementations . . . 38

4 Design of the Dataflow-Oriented ECGSoC 39
4.1 Introduction . . . 39
4.2 Digital Back-End Environment . . . 39
4.3 Streaming-Based Architecture Concept . . . 40
4.3.1 Streaming-Based Connection . . . 41
4.3.2 Reconfigurable Datapath . . . 41
4.4 Architectural Changes Applied to the ECGSoC . . . 42
4.4.1 Dataflow Within the ECGSoC . . . 42
4.4.2 Applying the Dataflow-Oriented Concept . . . 43
4.4.3 Proposed Dataflow Within the New ECGSoC Architecture 44
4.5 Algorithmic Changes . . . 45
4.5.1 Motion Artifact Removal . . . 45
4.5.2 Feature Extraction . . . 45
4.5.3 Beat Detection . . . 45
4.6 Accelerator Design . . . 47
4.6.1 Abstract Module Design . . . 47
4.6.2 Adaptive LMS Accelerator . . . 48
4.6.3 Dedicated FIR Filter . . . 48

5 Implementation of the dataflow-oriented ECGSoC 51
5.1 Introduction . . . 51
5.2 Tool Flow . . . 51
5.2.1 Synthesis Flow . . . 52
5.2.2 Application Performance Flow . . . 54
5.2.3 Power Estimation Flow . . . 55
5.3 Accurate R-peak Detection Application . . . 56
5.3.1 MatLAB Reference Model . . . 56
5.3.2 Architecture-Specific C . . . 56
5.3.3 Motion Artifact Removal . . . 57
5.3.4 Feature Extraction . . . 57
5.3.5 Beat Detection . . . 57
5.4 Dataflow-Oriented ECGSoC Processing Architecture . . . 58
5.4.1 SoC Top-Level Design . . . 58
5.4.2 Streaming-Based Communication . . . 59
5.4.3 Streaming Interface . . . 60
5.4.4 Input Data buffer . . . 62
5.4.5 Wrapper Generation . . . 63
5.4.6 Adaptive-LMS Accelerator . . . 64
5.4.7 FIR Accelerator . . . 64
5.4.8 Implementation Procedure . . . 65

6 Experimental Results **67**
6.1 Introduction . . . 67
6.2 ECGSoC V1.1 Evaluation . . . 67
6.2.1 DMA-Bus Power Evaluation Methodology . . . 68
6.2.2 DMA Bus Memory Evaluation . . . 69
6.2.3 Power Evaluation Results . . . 70
6.3 ECGSoC Algorithmic Verification . . . 73
6.3.1 Adaptive LMS Verification . . . 73
6.3.2 PCA Verification . . . 74
6.4 Physical Layout Results . . . 76
6.4.1 Synthesis Results . . . 76
6.4.2 Place and Route Results . . . 76
6.5 Power and Energy Consumption Results . . . 77
6.5.1 Power Consumption Overview . . . 77
6.5.2 Detailed Power Consumption . . . 78
6.5.3 Optimizations Overview . . . 79
6.5.4 Detailed Impact of Optimizations . . . 81
6.6 Experimental Results Summery . . . 84

7 Conclusions and Outlook **85**
7.1 Conclusions . . . 85
7.2 Future Work . . . 86

A Abbreviations & Symbols **90**
A.1 List of Abbreviations . . . 90
A.1 List of Symbols . . . 90

B Additional Information **91**
B.1 Comparison of Ultra Low-Power Processors for Sensor Networks . 92

List of Figures

1.1 The IMEC Human++ vision [18]. 2

2.1 ECG structure of a single heart beat, descriptive parameters ([82] with modifications). 7
2.2 PCA used for motion artifact removal [41]. 9
2.3 General concept of an adaptive filter [55]. 9
2.4 LMS used for motion artifact removal [66]. 10
2.5 CWT applied on a measured ECG [41]. 11
2.6 Typical sensor node architecture ([3] with modifications). 12
2.7 Power consumption in CMOS circuits. 13
2.7 (a) Dynamic power consumption. 13
2.7 (b) Leakage power consumption. 13
2.8 Clock gating methology [40]. 16
2.8 (a) Before clock gating. 16
2.8 (b) After clock gating. 16
2.9 Realistic power gating profile [40]. 16

3.1 Trade-off between energy efficiency and functional flexibility of different processing architectures (based on [88]). 20
3.2 GP and ASIP architecture [78]. 21
3.2 (a) Basic general-purpose architecture. 21
3.2 (b) Basic ASIP architecture. 21
3.3 Block diagram of a multi-core architecture [5]. 22
3.4 Block diagram of a SIMD Processor [63]. 23
3.5 Block diagram of a simple ASIC concept [78]. 23
3.6 Accelerators in SoC design [36]. 24
3.6 (a) Accelerator-rich architecture concept. 24
3.6 (b) Trade-off between energy and flexibility for accelerators. . 24
3.7 Ultra low-power processors for sensor networks in the last decade. 25
3.8 Evolution of on-chip communication architectures [60]. 27
3.9 Basic bus topologies [60]. 28
3.9 (a) Simple Bus. 28
3.9 (b) Hierachical Bus. 28
3.10 Full crossbar [60]. 29
3.11 Typical DMA configuration. 30
3.12 Router-based communication architecture [52]. 30
3.13 A system using socket-based interfaces [60]. 32

3.14 Power consumption breakdown of a simple bus-based communication architecture [49]. . . . 33
3.15 Mixed-signal ECGSoC and typical applications [41]. . . . 34
3.16 Algorithm overview. . . . 35
3.17 Blockdiagram of the digital back-end [41]. . . . 36
3.18 Block diagram of the ECGSoCs DMA-based communication [43]. 37

4.1 Basic concept of a dataflow-oriented hardware-accelerated processing architecture. . . . 40
4.2 Information transferred within a streaming channel. . . . 41
4.3 Visualization of the dataflow within the ECGSoC. . . . 42
4.4 Basic Concept of a dataflow-oriented approch applied to the ECGSoC. . . . 43
4.5 Proposed dataflow within the new architecture. . . . 44
4.6 R-peak detection algorithm. . . . 46
4.7 Adaptive threshold calculation. . . . 46
4.8 Streaming-based model design. . . . 47
4.9 Structure of a 4^{th} order adaptive NLMS algorithm. . . . 49
4.10 FIR structure with a folded delay line. . . . 49

5.1 Design tool flow overview. . . . 52
5.2 Synthesis tool flow. . . . 53
5.3 Application simulation tool flow. . . . 54
5.4 Power estimation tool flow. . . . 55
5.5 Top-level design of the proposed ECGSoC architecture. . . . 58
5.6 Signals necessary to implement streaming-based communication. 59
5.7 A single data transmission. . . . 60
5.8 Implemented functionality within the streaming interface. . . . 60
5.9 Data transfer across the asynchronous buffer. . . . 61
5.10 Input buffer overview. . . . 63
5.11 Code generator overview. . . . 63
5.12 The proposed ECGSoC floorplan including memories and power lanes. . . . 66

6.1 DMA bus consumption extraction tool flow. . . . 68
6.2 Subdivision of the memory power consumption. . . . 69
6.3 A single memory access. . . . 69
6.4 Top six consumers of the ECGSoC V1.1. . . . 71
6.4 (a) Adaptive LMS use case. . . . 71
6.4 (b) PCA use-case. . . . 71
6.5 Energy distribution of the ECGSoC, including the DMA bus. . . 72
6.6 Energy distribution of the data memory. . . . 73
6.7 Simulation results executing the adaptive LMS use case. . . . 74
6.8 Simulation results executing the PCA use case. . . . 75
6.9 The dataflow-oriented ECGSoC after the place and route stage. . 77
6.10 Average power consumption for the various optimization stages (adaptive LMS use case). . . . 79

6.11 Average power consumption for the various optimization stages (PCA use case). 80
6.12 Impact of the adaptive NLMS accelerator. 81
6.13 Impact of the dedicated FIR filter. 82
6.13 (a) Adaptive LMS use case. 82
6.13 (b) PCA use-case. 82
6.14 Impact of the proposed communication architecture. 83
6.14 (a) Total energy required for communication. 83
6.14 (b) Energy consumption caused by communicaiton overhead. 83

List of Tables

2.1 Sensor sampling rates of different phenomena (based on [22]). . . 6

6.1 Simulation conditions used for power evaluation. 70
6.2 Application duty-cycle for all use cases (ECGSoC V1.1). 70
6.3 Overall average power consumption. 71
6.4 Energy used for communication. 73
6.5 Area requirements of the proposed architecture as compared to the original system. 76
6.6 Area requirements of the proposed architecture after place and route. 77
6.7 Power consumption overview. 78
6.8 Power consumption details. 78
6.9 Accelerators impact on energy consumption. 82

B.1 Architecture comparison of ultra low-power processors for sensor networks. 92
B.2 Performance ratio comparison of ultra low-power processors for sensor networks. 95

Chapter 1

Introduction

1.1 Motivation

As a result of the advances in medicine, many diseases which were formerly lethal are nowadays chronic diseases. Furthermore, our populations are aging and living an increasingly unhealthy and stressful lifestyle. An outcome of this trend are the dramatically increasing healthcare costs within first-world countries. The costs of healthcare are highest when patients are hospitalized, although the person's quality of life in that scenario is at the lowest point. In contrast to that, treating patients in their own environment lowers the costs of healthcare and maximizes their quality of life [17].

The rapid advances in semiconductor integration and manufacturing provide the foundation for realizing the vision of ubiquitous healthcare. Small-scale wirelessly connected computing platforms forming body area networks, allowing long-term biomedical signal monitoring, are becoming reality [3]. Compared to the traditional hopital-centric healthcare system, remote healthcare monitoring enabling direct access to the patient's data by hospitals and medical doctors offers an attractive alternative [17].

Alongside the social and legal challenges, which have to be faced in order to take healthcare to the next level, various issues on the technology side have to be addressed. Issues such as low energy consumption, security reliability, as well as the system's form factor have to be solved [29]. The form factor especially has a direct impact on user acceptance. A high quality of life also presumes that the patient is not bothered by the on-body sensors. Battery size is significantly involved in or in some cases even defines the device's form factor. Therefore, the reduction of the sensor node's energy consumption is often declared as the highest-priority issue to address, since it directly influences battery lifetime as well as the device's form factor [29].

To reduce energy consumption or even reach energy autonomy, each system component of a sensor node has to be able to work within very tight consumption limitations in order to allow the overall system to perform within the available energy budget.

1.2 Wireless Sensor Networks for Biomedical Signal Monitoring

Wireless Sensor Networks (WSN) have been the focus of research over the past few decades, they have been marked as one of the most significant technologies in the 21st century [10]. It is, therefore, not surprising that these sensor networks have also found their way into the area of healthcare research.

Autonomous sensors remotely monitoring patients are a promising alternative to the traditional in-hospital monitoring, not only from an economic viewpoint but also from the patients perspective [3]. Various studies have shown, that patients would rather be treated in their own homes than frequently visit the hospital or stay in a health care facility [29]. Furthermore promoting the home-centric healthcare can help to reduce the drastically increasing costs of the healthcare system in the first world, and improve the life quality of our aging society [17].

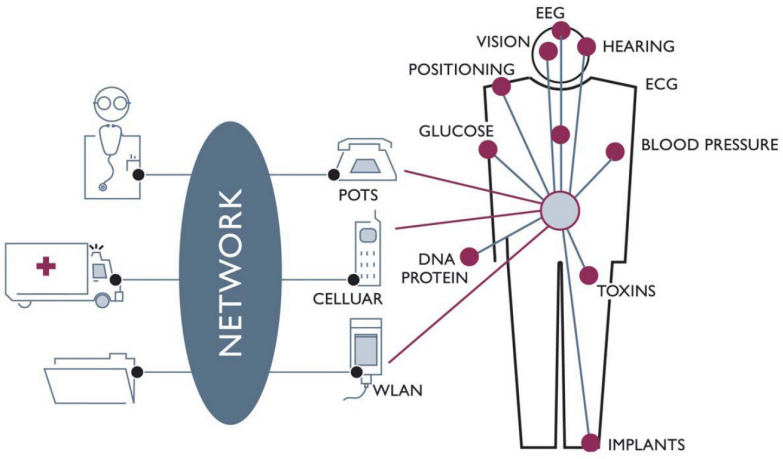


Figure 1.1: The IMEC Human++ vision [18].

Figure 1.1 depicts the IMEC Human++ Vision, where a set of small sensor/actuator nodes comprise a personal body area network (BAN), enabling medical supervision. These nodes are able of autonomously fulfilling their task, communicating with each other via the wireless channel and submitting their results to a base station. Furthermore, the network can transfer the gathered information to a medical database or the general practitioner using the traditional telecommunication infrastructure [18]. In a case of need the system could even place an emergency call, to notify the rescue services.

In the area of biomedical applications WSNs can provide improvements to the elderly care, help managing chronic diseases and enhance rehabilitation, wellness and lifestyle monitoring [3]. To enable the use of WSNs for healthcare monitoring, various technological, legal and social challenges have to be faced [29].

1.3 Thesis Goals

The following main goals are targeted by this master thesis in order to develop a dataflow-oriented processing system which is suitable for the requirements of a wireless sensor node applied in biomedical applications.

- Evaluation of the baseline system:
  - Identification of whether the if algorithms are suitable for an accelerator-based implementation.
  - Analysis of communication architecture in order to identify optimization potential.
- Dataflow-oriented architecture design:
  - Design of a suitable communication architecture for dataflow-oriented hardware-accelerated architecture.
  - Definition of a generalized module structure in order to allow a reconfigurable communication structure.
  - Support of various low-power techniques.
- Implementation of the optimizations:
  - Enabling of operating frequency-independent communication.
  - Support of low-power states of various components.
  - Increase the designs power efficiency.
- Synthesis and place and route of the architecture design:
  - HDL implementation of the design.
  - Synthesis and place & route to achieve a physical layout.
  - Estimation of the placed and routed system's power consumption.
- Verification and evaluation:
  - Verification of the simulation results based on a reference model.
  - Impact evaluation of the applied optimizations.

1.4 Thesis Organization

In Chapter 2 an overview of Wireless Sensor Nodes, their general architecture, usage in society and biomedical applications is given. Furthermore, the fundamentals of power consumption in CMOS circuits are provided. Chapter 3 discusses the various processor architectures as well as on-chip communication architectures applied in sensor nodes. Moreover, the baseline system and previous work on ECG monitoring systems is presented. Chapter 4 gives insight on the design process of the hardware and software. The concept of the architecture as well as the generic module design are described. The actual implementation including the applied optimizations is outlined in Chapter 5, as well as, the used tool flow. Chapter 6 discusses the system verification and the obtained physical layout and power results for the implemented architecture. Finally, Chapter 7 concludes the thesis and provides ideas and suggestions for future work.

This page is intentionally left blank.

Chapter 2

Fundamentals of Sensor Nodes and Medical Applications

2.1 Introduction

The origin of WSNs can be traced back to a research program named Distributed Sensor Networks [10], a time when researches started facing the challenge of performing necessary computation, limited by the provided possibilities of a given sensor platform [22].

The idea of a sensor network has not changed since then, the basic building blocks of every WSN are the individual nodes. Each sensor node is capable of sensing, computing and communicating wirelessly as an independent system. Combined in a network, these embedded systems are useful in a wide range of applications.

2.2 Wireless Sensor Networks in Our Society

Our society already uses sensor networks for various tasks. Due to the fact that these arrays of sensor nodes allow data gathering and analysis in a way traditional instrumentation can not carry out these tasks [81], sensor networks are applied in various areas of our daily lives.

Use cases like infrastructure security or environment monitoring are safeguarding our society unknowingly, as in volcano monitoring [81] or an early-warning-system for mass-movement [75]. Sensing networks are able to improve the quality of various industrial applications, because sensing is one of the cornerstones in many ares of industrial automation [45]. Furthermore, they can increase our personal comfort-level when they are applied in medical healthcare [3].

Table 2.1 shows various physical phenomena categorized by the necessary monitoring sample frequency. These physical events are the basis for various possible applications. The sample rate required to acquire the needed information has an important influence on the computational capabilities of the node itself.

Table 2.1: Sensor sampling rates of different phenomena (based on [22]).

Phenomena	**Sample Rate** (in Hz)
Very Low Frequency	
Atmospheric temperature	0.017 - 1
Barometric pressure	0.017 - 1
Low Frequency	
Heart rate	0.8 - 3.2
Volcanic infrasound	20 - 80
Natural seismic vibration	0.2 - 100
Mid Frequency	*(100Hz - 1000 Hz)*
Earthquake vibrations	100 - 160 Hz
ECG (heart electrical activity)	100 - 250
High Frequency	*(> 1kHz)*
EEG (brain electrical activity)	200 - 2k
EMG (muscle electrical activity)	1k - 2k
Breathing sounds	100 - 5k
Industrial vibrations	40k
Audio (human hearing range)	15 - 44k
Audio (muzzle shock-wave)	1M
Video (digital television)	10M

Classical tasks executed on a sensor node frequently behave repetitively, with the aim to respond to external or internal events. In general sensor node applications are inherently event-driven [24]. The computation workloads occurring in a WSN are typically more varied and application-dependent compared to classical workload measures like a CPU benchmark. Most use cases require a combination of idle and active computation [22]. The sample frequency is highly correlated to the system idle and active times. Furthermore, the active computation time effects the power budget of the system, so the phenomena has a large influence on the systems requirements.

2.3 Medical Applications for Wireless Sensor Nodes

Section 1.2 indicated that WSN can be used for various medical monitoring or therapeutic purposes. Sensor networks enable long-term continuous monitoring in a way, today's devices are not capable of carrying out. Having more complex systems will allow detection or even prediction of emergency situations. Furthermore, long-term recording can be helpful in rehabilitation or physical training progress [3].

An enabling prerequisite for the improvement of biomedical monitoring for WSNs is the ability of a sensor node to capture physical signals related to the health status of an individual. Examples of biomedical signals a node could

capture would be Electro-Encephalogram (EEG), which enables diagnosis of sleeping disorders or epilepsy, or Electrocardiogram (ECG) to diagnose heart arrhythmia [17]. Moreover, Electromyography (EMG) enables detection of neuromuscular diseases [3], and even advanced concepts like stress monitoring are imaginable [17].

Physiological signals like EEG and ECG are basic parts of many biomedical applications and diagnoses. Especially ECG is a non-invasive method representing an individual's medical condition [3].

2.4 Electrocardiography

The heart itself fulfills the life-sustaining task as an actuator for our blood circulation. The timing and coordination of the myocardial muscle is the responsibility of the heart's electrical conduction system. Due to the fact that the ECG is a non-invasive method, it is an important technique for diagnostic investigation [30]. Figure 2.1 depicts an ECG corresponding to a single heart beat.

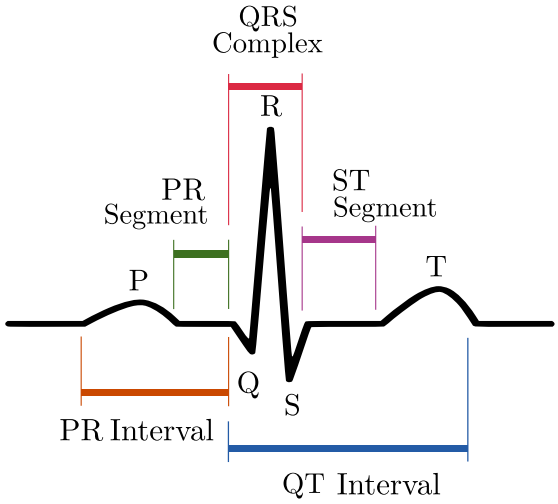


Figure 2.1: ECG structure of a single heart beat, descriptive parameters ([82] with modifications).

Furthermore it shows the important parameters necessary for a quantitative representation of the signal. For example, the QRS-complex characterizes the electrical excitation of the working musculature in both ventricles. The distance between two R-peaks is the foundation for calculating the heart rate, commonly known as pulse. Based on all these parameters, a medical professional is able to recognize various diseases [30].

Breathing, movement and muscle activity are the common interferences in ECG [30]. This noise is caused by the movement of the ECG electrode, and commonly identified as motion artifact (MA). These artifacts can affect the interpretation of an ECG, in case of large interference the unwanted signal can be very similar to P and T waves as well as the QRS-complex. This can lead

to misinterpretation of the signal and therefore, can result in inappropriate treatment and misdiagnosis [77].

There are two bu challenges concerning ECG monitoring performed by a sensor node. First, minimizing the influence of motion artifacts, which are expected to be higher due to more body movement in ambulatory supervision as compared to a clinical monitoring environment. Second, automatic analysis of ECG signal with limited computational performance [65].

2.4.1 Motion Artifact Removal

Several techniques were proposed to address the challenge of motion artifact removal in ECG. Methods using wavelet transform [44] or adaptive filters [41], as well as blind source separation [66] were introduced. The following section is going to introduce adaptive filtering and principal component analysis (PCA), which are the applied motion artifact removal techniques in this thesis.

2.4.1.1 Principal Component Analysis

PCA is a blind source separation technique, which can be applied to reduce motion artifacts, due to the fact that the ECG signal and the interfering artifacts are uncorrelated [64].

The general idea behind PCA is reducing the dimension count of a data set, while retaining as much information as possible. To achieve this, the interrelated data is transformed into a new set of linearly uncorrelated variables called the principal components. The first few of these components represent the most common information presented in the original data set [38].

$$\mathbf{Y} = \mathbf{\Psi X} \tag{2.1}$$

Equation 2.1 represents the central concept of PCA, where $\mathbf{X}$ is the original data set. The dimensionality of $\mathbf{X}$ is defined be the number of experiments and the corresponding sample count. $\mathbf{Y}$ represents the resulting principal components, which can be computed with the help of an orthogonal linear transformation matrix $\mathbf{\Psi}$ [9]. For further details and different methods for deriving transformation matrices for PCA the interested reader is referred to [38].

To use PCA for motion artifact removal, it is necessary to collect the data of n statistical independent ECG leads. Applying the PCA onto the data will result in n principal components, where the first component represents the highest variance, and the *n-th* the lowest [64]. Assuming small motion artifacts, the high variance component represents mainly ECG information, while the *n-th* component corresponds to the interference [9]. In ambulatory monitoring conditions, the allocating principal component can be more complex, due to stronger influence of motion artifacts [64].

Figure 2.2 depicts a PCA applied to a 3-lead ECG signal. Three channels were used to collect the statistically independent ECG signals, all these channels contain motion artifacts in various orders of magnitude. The waveform on the bottom shows the cleaned ECG signal after applying the PCA algorithm. The gray highlighted sections depict the influences of motion artifacts, a comparison

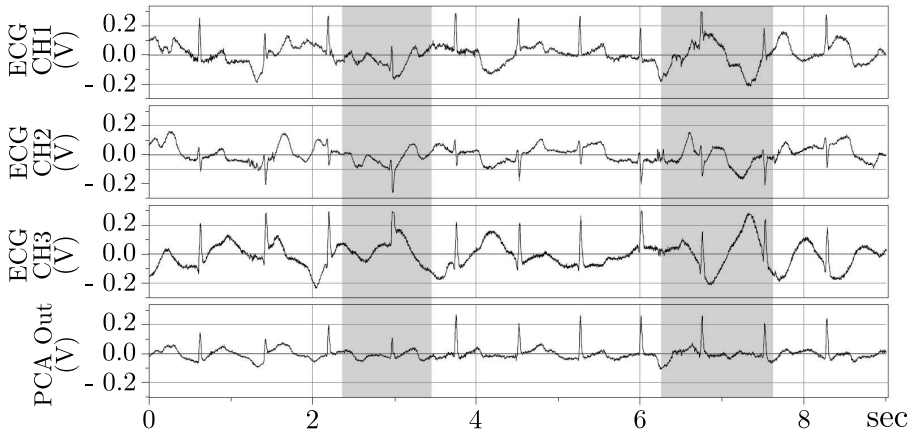


Figure 2.2: PCA used for motion artifact removal [41].

between the different channels shows that the interferences can have different magnitudes and polarities.

2.4.1.2 Adaptive Filter

The general concept of an adaptive filter is depicted in Figure 2.3. For each sample of the input signal, the adaptive filter calculates the corresponding output ($y[k]$). The difference between the desired signal ($d[k]$) and the result of the filter produces the error signal ($e[k]$). Based on the error, the adaption algorithm corrects the fault by modifying the filter coefficients. Adopting the filter coefficients enables the system to learn the linear correlation between the input and the desired signal. An adaption algorithm is generally designed to minimize the mean square error [55].

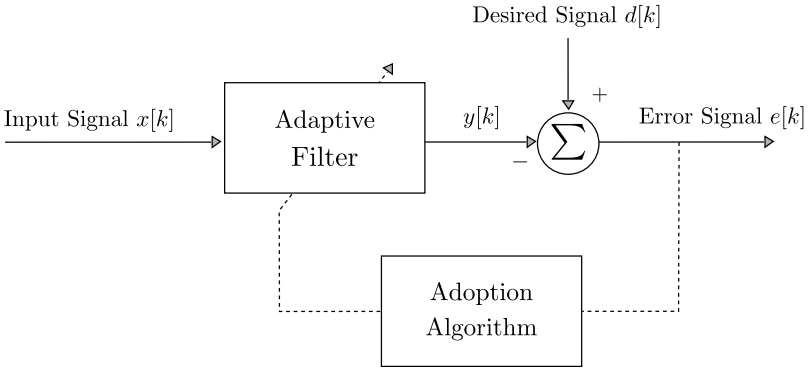


Figure 2.3: General concept of an adaptive filter [55].

The most commonly used adaption algorithm for finite impulse response (FIR) adaptive filters is the least-mean-square algorithm (LMS). The algorithm is often preferred for its simplicity. Equation 2.2 shows the necessary compu-

tation for updateing the filter coefficients. The vector $\vec{w}$ represents the filter coefficients and the scalar factor μ controls the step size of the adoption [55].

$$\vec{w}[k+1] = \vec{w}[k] + \mu e[k]\vec{x}[k] \tag{2.2}$$

To assure convergence of the LMS algorithm the step size has to be in a defined range. This range given in Equation 2.3 shows that the theoretical maximum is defined by the highest Eigenvalue of the input signal.

$$0 < \mu < \frac{2}{\lambda_{max}} = \mu_{max} \tag{2.3}$$

Equation 2.4 presents a more conservative and practical approach for finding the step size's upper bound. The expectation of $|x[k]|^2$ represents the input signals average power.

$$0 < \mu < \mu_{max2} = \frac{2}{N \cdot \mathbb{E}\{|x[k]|^2\}} \tag{2.4}$$

For further background on adaptive filters, the interested reader is referred to [55].

As Section 2.4 indicated that the electrode movement is the main source of motion artifacts, more precisely the changing electrode-tissue impedance causes the interferences. To reduce the effect of motion artifacts, therefore, the measured impedance can be fed to an adaptive filter as a reference signal [66].

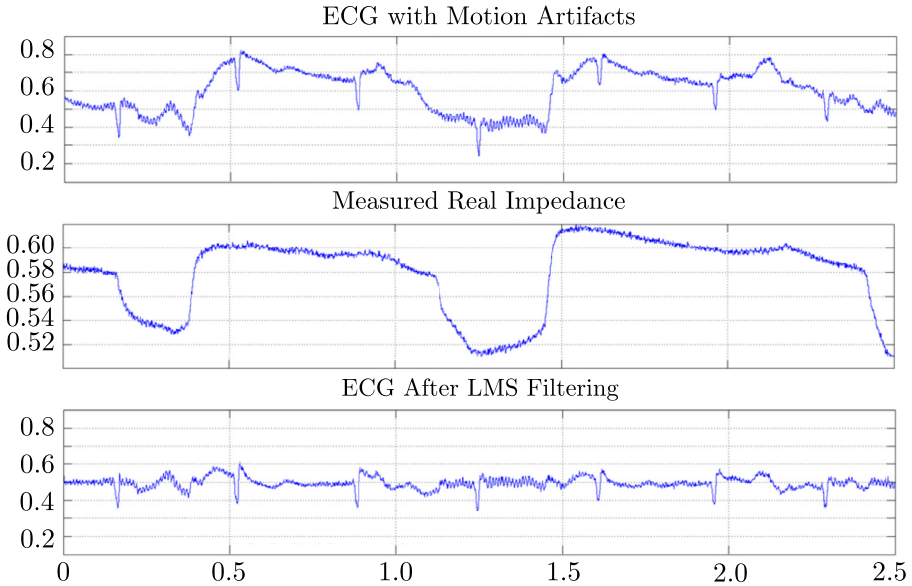


Figure 2.4: LMS used for motion artifact removal [66].

Figure 2.4 depicts an adaptive LMS filter used for motion artifact removal. It shows the high correlation between the noisy artifacts and the measured electrode-tissue impedance. The input signal of the system is the noisy ECG signal, using the impedance of the electrode as the desired signal, the adaption algorithm is able to remove the motion artifacts from the gathered ECG.

2.4.2 Automatic ECG Analysis

Various algorithms have been proposed to target one of the important challenges in automatic ECG analysis, the beat detection. The methods ranged from simple deviation based algorithms to Neural Networks [65]. The next section introduces the Continuous Wavelet Transform (CWT), which is used within this thesis.

2.4.2.1 Continuous Wavelet Transform

The CWT is able to localize high frequency signal features in time, by using a flexible Heisenberg box. This allows associating the window size to the observation. This, and the fact that CWT also allows usage of non-sinusoidal analysing functions, are the main differences between the CWT and the Short Time Fourier Transform [51].

$$T(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} x(t)\psi^* \left(\frac{t-b}{a}\right) dt \tag{2.5}$$

Equation 2.5 defines the wavelet transform for a continuous time signal $x(t)$. The complex conjugated wavelet function is $\psi^*(t)$, and a and b are parameters to describe the dilation and location of the wavelet function. A wavelet has to satisfy certain mathematical criteria, therefore, a large selection of waveforms can be employed with CWT [51]. For further details, the interested reader is referred to [51].

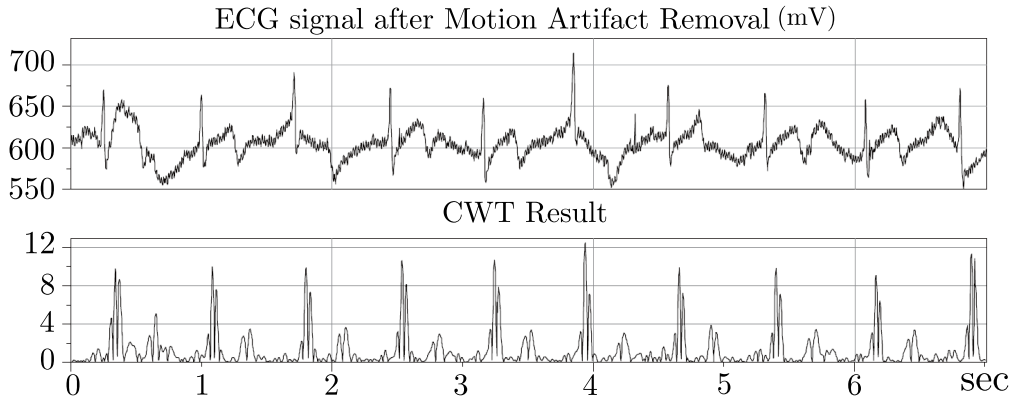


Figure 2.5: CWT applied on a measured ECG after motion artifact removal [41].

Methods based on the modulus maxima wavelet are increasingly used to analyse signals in medical and engineering applications [51]. Usage of the "Mexican Hat" wavelet allows very robust beat detection of ECG signal [65]. Figure 2.5 depicts the result of an CWT, using the "Mexican Hat" wavelet, applied to a ECG signal where the motion artifacts are already removed. It shows that the highest peaks in the resulting waveforms correspond to the beats in the ECG signal.

2.5 Typical Wireless Sensor Node Architecture

A sensor node is comprised of multiple components, which are able to handle digital, analog and mixed signals. Figure 2.6 depicts the main building blocks of a generalized node architecture. Sensors are capable of converting physical or chemical quantities, into processable electrical signals. The analog front-end (AFE) amplifies and conditions the signals from the sensor to reduce the noise and improve the signal-to-noise ratio. In case the node can control external components, or is even able to influence the monitored phenomena, actuators are used to influence the node's environment. To allow further processing of the gathered signals, they have to be converted into the digital domain, this signal conversion is performed with the help of analog-to-digital converters (ADC). Depending on the node's use case, the system comprises a microcontroller and/or a digital signal processor (DSP). A microcontroller is included to perform control-

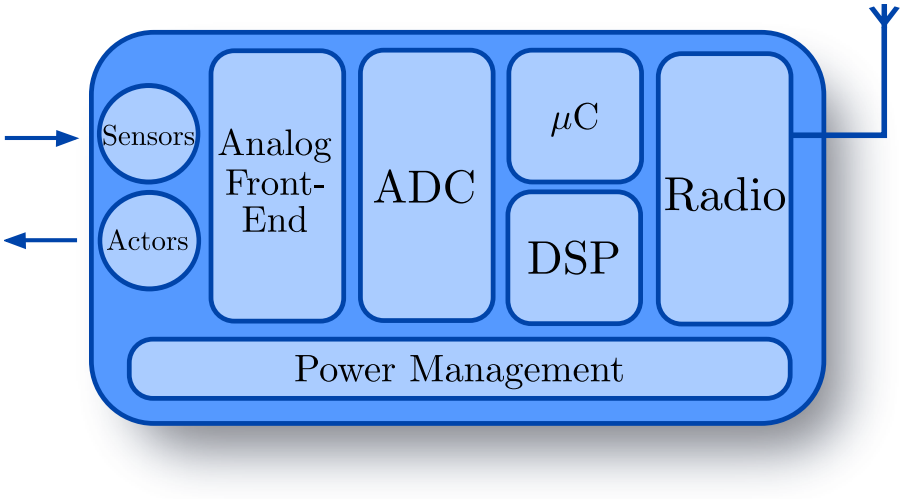


Figure 2.6: Typical sensor node architecture ([3] with modifications).

related tasks and, in case of small captured data amounts, it is also capable of processing the sensor signals. On the other hand, the DSP is able of processing larger amounts of data and performing high-speed calculations. The sensor node transmits either extracted features or the complete reconstructed signal using the radio module to a base station. To enable the embedded system to stay within its limited energy budget and avoiding instantaneous power consumption peaks, the node is equipped with a power management (PM) unit. Existing energy storage and/or energy harvesting devices are limiting the long-term energy budget for a sensor node, therefore the PM considers all other components of the system and supervises their active and idle states.

The main power consumers on a classical node architecture are the processing unit as well as the radio module used for data transmission. Especially in terms of power consumption, a trade-off between these two components exists [3], i.e. the more data has to be transferred via the wireless channel, the

more power is necessary to supply the radio unit. In contrast, the energy, used to reduce the amount of data by extracting features and compressing information, increases due to the increasing amount of computation performed by the processing unit.

In a typical WSN application up to 50% of the total power consumption can be spent supplying the processing unit and providing the necessary computational performance [23]. Furthermore, in case of a continuous data stream, i.e. where marginal computation is necessary, the radio power consumption can even exceed the processing power consumption by a factor of 10x [3].

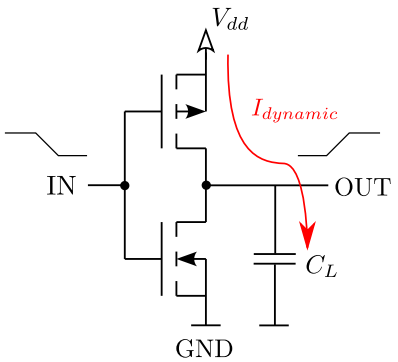
As indicated in Section 2.2 it can be seen, that the amount of data gathered by the system has a high influence on the resulting power consumption of the sensor node. The application, more precisely, the required sample rate to monitor the physical quantities represents an important factor for designing the node architecture. This implies that according to the application, the emphasis of the consumed power will vary between the components of the architecture.

2.6 Sources of Power Consumption and Optimization Techniques

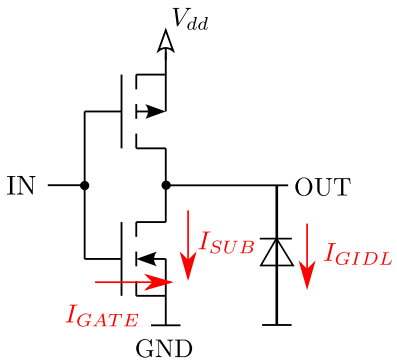
In CMOS circuits the power consumption is subdivided into two main contributors: *static* power P_{static} and *dynamic* power $P_{dynamic}$ [39]. Equation 2.6 shows that the sum of these two components expresses the total power dissipation.

$$P_{total} = P_{dynamic} + P_{static} \tag{2.6}$$

When a device is active, more precisely a value is changing, dynamic power is consumed. On the other hand, static power is consumed while a powered-up device is in idle state, without changing values. Leakage is the main source of static power dissipation in CMOS devices [40].



2.7 (a): Dynamic power consumption.



2.7 (b): Leakage power consumption.

Figure 2.7: Power consumption in CMOS circuits.

2.6.1 Dynamic Power Consumption

Dynamic power consumption has been the primary source of power consumption for many years. The dominant source is the power required to charge/discharge the capacitance connected to the gate's output [39].

$$P_{dynamic} \propto C_L \cdot V_{dd}^2 \cdot A \cdot f_{Clk} \tag{2.7}$$

Equation 2.7 expresses the dynamic power caused by switching, where V_{dd} is the supply voltage, f_{Clk} the clock frequency and C_L the load capacitance. The activity factor A describes the probability of an output transition. It can be seen, that the switching power is a function of load capacitance and the device activity and therefore data-dependent [40]. Figure 2.7 (a) depicts the charging current causing the switching power.

The internal switching current, required to charge the cell's internal capacitance, as well as the short circuit current, occurring while both transistors are transitioning, also contribute to the dynamic power consumption. Assuming short input signal ramps, the dynamic power is dominated by the power used to charge/discharge the load capacitance. As Equation 2.7 shows, the most effective way is to reduce the supply voltage. Due to the quadratic contribution to the switching power scaling V_{dd} was used over the last decades to control dynamic power dissipation [40].

2.6.2 Static Power Consumption

While the dynamic power consumption was the limiting factor for many years, the leakage current is getting increasingly dominant in recent technologies [39].

In contrast to our idealized view of a MOS transistor, which acts like a switch, the transistor is an analog device and is affected by sub-threshold and leakage currents.

$$P_{static} = V_{dd} \cdot (I_{SUB} + I_{GATE} + I_{GIDL} + I_{REV}) \tag{2.8}$$

Figure 2.7 (b) depicts the main contributors to the leakage currents. The Sub-threshold leakage I_{SUB} occurs when the transistor is in weak inversion between drain and source. Gate Leakage I_{GATE} is caused by hot carrier injection and gate oxide tunneling. The high field effect-induced current Gate Induced Drain Leakage I_{GIDL} and the Reverse Bias Junction Leakage I_{REV}. The total leakage current is the sum of all mentioned components, as expressed in Equation 2.8.

$$I_{SUB} = \mu C_{ox} V_{th}^2 \frac{W}{L} \cdot \exp\left(\frac{V_{GS} - V_T}{n \cdot V_{th}}\right) \tag{2.9}$$

Equation 2.9 shows a good approximation for the sub-threshold leakage current, where W and L are the according transistor dimensions, V_{th} the thermal voltage and the fabrication process parameter n. It expresses an exponential dependency between the leakage power and the difference between gate-source and threshold voltage.

This leads to a conflict when the V_{dd} is scaled, i.e. lowering the supply and threshold voltage to reduce dynamic power consumption results in a increasing

leakage power dissipation. In technologies above 90nm the leakage is much smaller than dynamic power, but beneath static power consumption it can be in the same order of magnitude as the dynamic power dissipation [40]. The seriousness of increasing leakage current in younger process technologies, where the off-current is raised by a factor of 10 for each generation, as described in the ITRS 2011, where reducing the leakage in CMOS is defined as a long-term goal [35].

2.6.3 Low Power Techniques

Over the last decades various low power optimization techniques were developed to reduce dynamic and static power. Designers are forced to use other techniques than technology scaling, because the energy savings provided by scaling are decreasing [19].

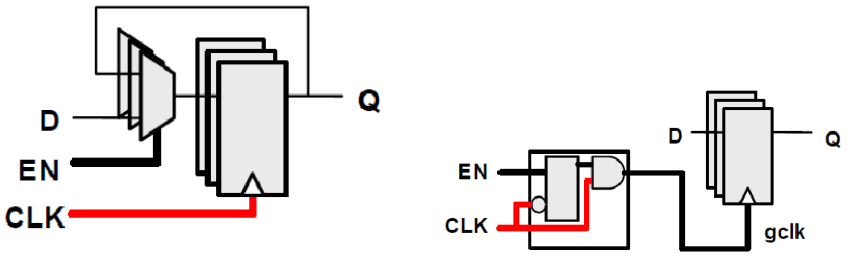
Considering the main influences on dynamic power consumption as expressed in Equation 2.7 the following four variables should be minimized:

- **Supply Voltage** (V_{dd}): The most effective method to reduce the dynamic power consumption is to lower the supply voltage. But on the other hand it necessities to lower the threshold in order to maintain performance, which increases the leakage current.
- **Clock Frequency** (f_{Clk}): Due to the far-reaching and fundamental impact on a system's power consumption, minimizing the clock frequency is a traditional low-power method.
- **Load Capacitance** (C_L): Reducing the output load of a gate can be achieved by minimizing the wire length and the number of driven gates.
- **Activity Factor** (A): Minimizing the transition probability also has a high impact on the power dissipation. E.g. using encoding techniques or reducing the cycle count of an instruction can reduce the system's power consumption.

The following paragraphs describe state-of-the-art low-power-optimization techniques which can be applied to reduce a systems power dissipation.

Clock Gating is the most common way to reduce the power dissipation, as it reduces the activity factor by turning off the clock signal, if it is not required. The so-called clock tree, the distribution network for the clock signal, can consume up to 50% of a system's dynamic power [40]. Without *clock gating* all cells would be constantly clocked and therefore consume power, so gating the clock signal is a logical step to reduce the system's toggle rate.

Figure 2.8 (a) depicts a classical non-gated register, where the high activity clock signal toggles each flip-flop every clock cycle. After *clock gating*, the registers are only supplied with a clock signal in case of an activated enable signal, as shown in Figure 2.8 (b). More precisely the high activity clock signal is gated by the clock gating cell. This method is only successful when the number



2.8 (a): Before clock gating. 2.8 (b): After clock gating.

Figure 2.8: Clock gating methology [40].

of gated cells is higher than the necessary gate cells, otherwise no power would be saved. Current design tools are able to automatically insert clock gates without influencing the logic functionality of the design.

Power savings of up to 40% can be acquired for a processor in idle mode using *clock gating* [40].

Power Gating is a technique which allows to reduce the overall leakage consumption of a system. Especially in newer CMOS technologies this method is highly desirable due to the exponentially increasing sub-threshold leakage current mentioned in Section 2.6.2.

The method is as simple as it is effective: by turning off the supply voltage for blocks which are not used, while active blocks are staying supplied, it reduces the system's leakage consumption. This, of course, has its drawbacks, *power gating* is highly invasive, effects on-chip communication and can lead to timing delays [40].

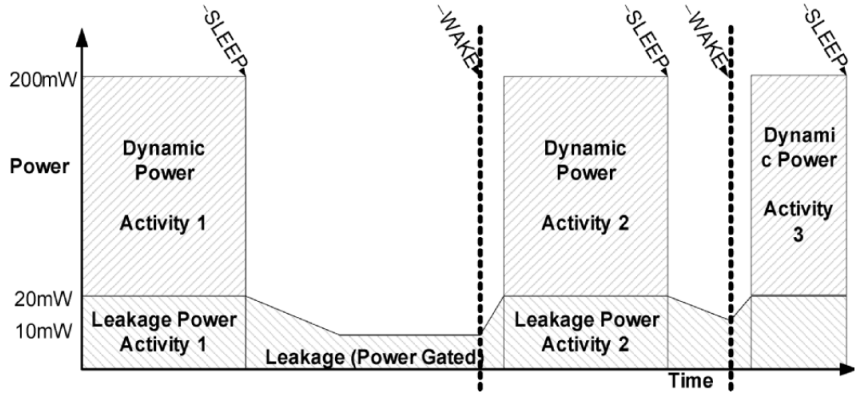


Figure 2.9: Realistic power gating profile [40].

Figure 2.9 depicts a typical application where *power gating* can be applied. Dynamic power reduction methods like clock gating and gating the supply voltage between activities allows to reduce the systems leakage consumption in the SLEEP state. The delay added due to *power gating* is shown by the difference

in time between the WAKE signal and the activity start time.

While *power gating* whole blocks is a very attractive solution, the impact on general-purpose processors is limited [22]. Furthermore, the fine-grained *power gating* of a VLIW processors datapath can produce more overhead than it reduces consumption [58].

Operand Isolation is used to prevent the combinatorial logic from unnecessary operations, when the block is not active. Therefore a so-called guard logic is added between the data input and the combinatorial logic. In the literature this method is known as *value gating* or *guarded evaluation* and reduces dynamic power by avoiding switching in blocks which do not carry significant information [39].

The Multi-Threshold Logic approach is able to optimize timing and reduce leakage, by choosing the optimal combination of transistors with differing threshold voltage (V_T). Nowadays vendors of standard cell libraries offer a selection of the cells with different thresholds. The threshold has a high influence on the gate delay, and so the synthesis tool is able to optimize the system to meet the timing and power constrains [40]. In addition, this method can also be used to reduce the leakage current in the power switches used for power gating [67].

Multi-Voltage design is based on the idea of using different supply voltages for various partitions of a chip, called *power domains* or *voltage islands*. This allows supplying each system component in regard to its own performance requirements and, therefore, can provide relevant power reduction. Due to the influence on the gate delay, when the supply voltage is lowered, the supply of components impacts system performance [40].

Dynamic Voltage Scaling (DVS) is basically an extension of the multi-voltage approach, which allows to adjust the voltage levels dynamically. DVS enables a system to adopt a component's supply voltage and adjust the maximum clock frequency accordingly to the application's specific needs. This technique is already state-of-the art in various commercial vendors processors [28]. Furthermore Ultra-Dynamic Voltage Scaling (UDVS) allows voltage scaling in a range from sub-threshold to above-threshold [79].

Sub-Threshold Design is an approach where transistors are supplied with a voltage beneath the component's threshold voltage, more precisely, the transistors operate in weak inversion. Operating in this mode, the device consumes less power when active and less leakage power when not. On the other hand, transistors in weak inversion operate much slower compared to devices with higher supply voltage [79].

In an application where computation speed is less of a goal than energy efficiency, like some WSN applications, different processors operating beneath the transistor threshold voltage were proposed [86, 56, 21].

Asynchronous Systems are circuits which are able to fulfill their task without a presence of a clock signal. Therefore a major source of power consumption, the clock tree, is eliminated. Furthermore, asynchronous systems cannot operate without any input data, so a system in idle state will not consume dynamic power [69].

A subcategory of asynchronous systems are globally asynchronous/locally synchronous (GALS) devices. Such designs are able to solve global timing issues on large dies, where the signal propagation delay can be a problem. The idea behind GALS is to use classical clocked systems in local regions of the chip and use an asynchronous design on system-level [14].

Chapter 3

Processing Systems in Wireless Sensor Nodes

3.1 Introduction

As mentioned in Section 2.5, the processing unit is a main contributor to the system's power consumption. Over the last few years various processors were applied in sensor nodes, from off-the-shelf components to highly specialized ultra-low-power (ULP) systems consuming less than one picojoule per instruction [22].

To enable computation at such low power consumption, the system has to be specialized for the specific sensor node workload. A holistic approach is necessary to allow such low power performance and to meet the lifetime requirements of a WSN application [23]. Furthermore, the International Technology Roadmap for Semiconductors (ITRS) predicts that the behavioral and architectural power minimization techniques will be more important than traditional physical and register-transfer-level methods within the next few years [35]. Therefore this thesis focuses on two main categories of actual SoC design, namely:

- **Processor Architectures** and
- **On-Chip Communication Architectures**

Furthermore, it discusses a possible way to combine these two design challenges and solve the issues from an architectural point of view.

This chapter covers various aspects of processing unit design for low power applications. First, an overview of processor architectures and design trends of the last decade is presented. Second, on-chip communication architectures and interconnect systems are described. Finally, a low-power sensor node processing system, namely the ECGSoC is shown, the system which is the foundation of this thesis.

3.2 Processor Architectures in Wireless Sensor Nodes

Various different architectures for processing units exist, every implementation type has its advantages and drawbacks. A way to compare these different sys-

tems is shown in Figure 3.1; it depicts the trade-off between the energy efficiency and the functional flexibility of state-of-the-art architectures. While dedicated hardware needs a minimum of energy to fulfill the task, it is not able to perform computations other than the ones it was designed for. On the other hand, a

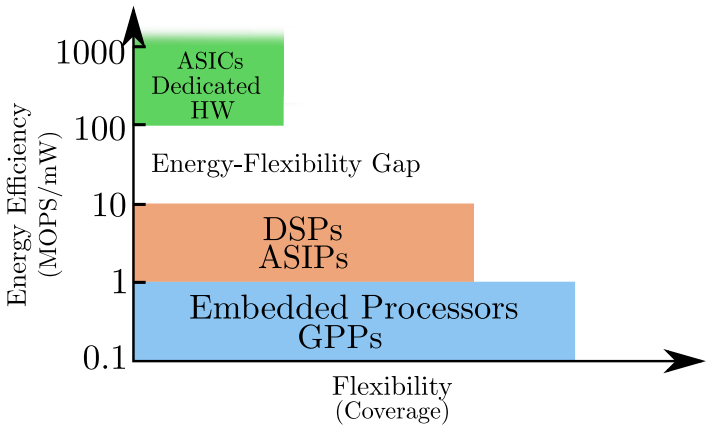


Figure 3.1: Trade-off between energy efficiency and functional flexibility of different processing architectures (based on [88]).

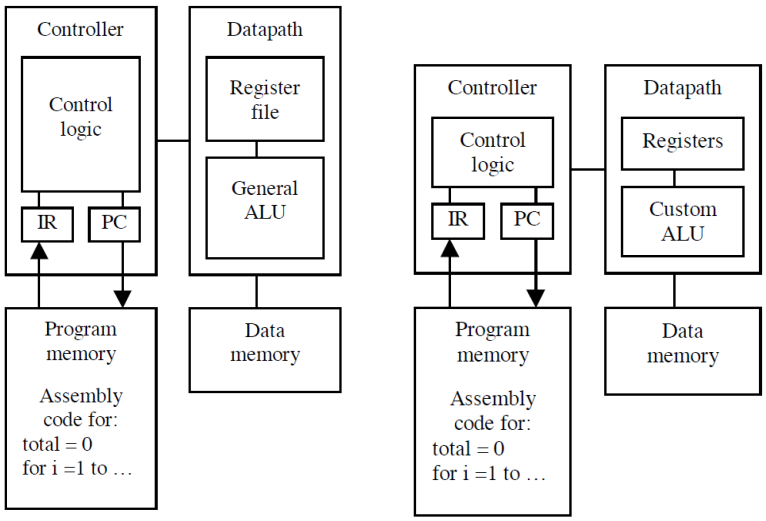
general purpose processor is completely adoptable via software, but less power-efficient. The energy efficiency is up to a factor 500x worse compared to a Application Specific Integrated Circuit (ASIC) [19].

In the last decade, reconfigurable processors [8, 88] tended to fill the energy-flexibility gap between dedicated hardware and a class of Application Specific Instruction-Set Processors (ASIP). Nowadays, a newer form called Accelerator-Rich Architectures [36] also has the potential to close this gap.

The following sections will give a short overview about the existing architecture types, while focusing on the applicability as a processing unit for a sensor node.

3.2.1 Processor Architectures

General Purpose Architectures also known as Central Processing Unit (CPU) represent the most common processor architecture. The architecture is designed to execute computation tasks for a large range of different applications, as the name indicates. As depicted in Figure 3.2 (a) a general purpose processor (GPP) uses a datapath, with access to the data memory, which is actuated by the control logic. More precisely the controller executes the fetched instructions from the program memory in the datapath, accordingly to the current program counter (PC) [78].



3.2 (a): Basic general-purpose architecture.

3.2 (b): Basic ASIP architecture.

Figure 3.2: GP and ASIP architecture [78].

Application Specific Instruction-Set Processors are processors designed for a particular application or a set of applications with common characteristics. To meet the desired power requirements and performance an ASIP is specialized to fit the computational nature of an application [12]. Therefore, a specialized instruction-set is implemented, i.e. various custom instructions are supported by the *Arithmetic-Logic-Unit* (ALU). Figure 3.2 (b) illustrates that the custom ALU is the fact that distinguishes an ASIP from an GPP. The customization of the datapath also allows elimination of infrequently used units, which helps to reduce the platform's power consumption and size [78].

Digital Signal Processors are highly demanded these days and used in various applications where digital-signal processing is necessary. DSPs are basically a commonly accepted class of ASIPs, designed to meet the needs of various math-intensive computations. They often include hardware components like a multiply-accumulate unit, which is a very common operation for filter implementations. A separate memory interface, which allows sequential data fetching in parallel with other instructions to improve performance on large data arrays, is often implemented in a DSP [78] as well.

Asynchronous Processors are whole processing systems implemented in an asynchronous fashion, i.e. no clock signal is present in the whole processor. The advantages of asynchronous systems were already discussed in Section 2.6.3. At this point is has to be mentioned, that in asynchronous processors the circuit style, more precisely the protocol and data encoding, has a high influence on the system's power consumption [69].

Multi-Core Architectures have already become commonly used, and are present in most current personal computers. The architecture combines multiple complete processing units including caches on a single chip [5]. Depending on the purpose of the multi-core processor, the shared memory might be directly placed on the chip or accessible through an external interface. Such systems are called *shared memory multiprocessors* (SMP) and offer a physical address space for all cores [61]. Figure 3.3 depicts the concept of a typical dual-core processor. Multiprocessor architectures where each core has its own unique address

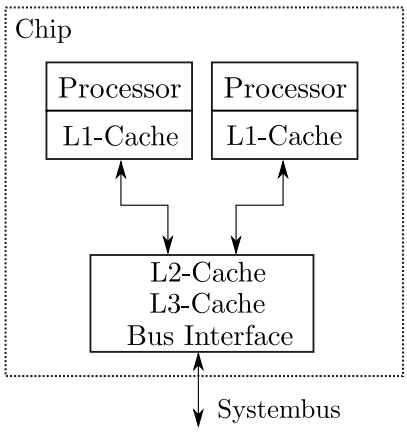


Figure 3.3: Block diagram of a multi-core architecture [5].

space and is connected to the other processing units via an interconnection network, are named *message-passing multiprocessors* or clusters. These systems are only able to communicate by sending and receiving messages, so-called message passing [61].

Nowadays multi-core processors can already be found in various embedded systems and also have found their way into commercial digital signal processing [34].

Single Instruction Multiple Data and Vector Architectures are designs which are able to execute instructions on data vectors. A typical *Single Instruction Multiple Data* (SIMD) supporting architecture is shown in Figure 3.4. As depicted, a SIMD instruction can be executed within one cycle, by sending multiple data to the according number of *processing elements* (PE) [61]. To support SIMD features, a common design is to add a separate vector datapath including registers and a separate vector ALU, besides the existing scalar datapath. Especially in digital signal processing, where the algorithms are commonly processing large amount of data, SIMD is able to increase the computation performance [63].

ASIC or Single-Purpose Processors are highly specialized designs developed to compute a specific task, therefore these devices offer limited flexibility. Such a processor is a custom digital circuit implemented as dedicated hard-

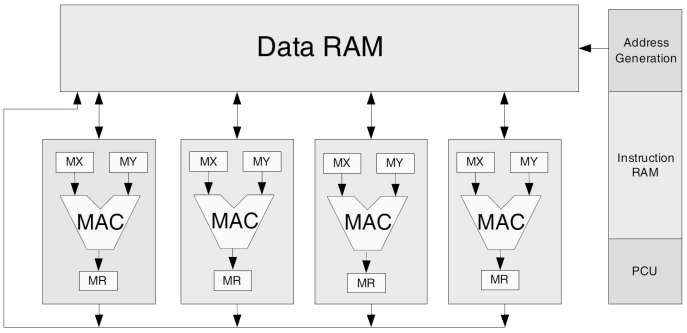


Figure 3.4: Block diagram of a SIMD Processor [63].

ware, more properly known as an ASIC [78]. In terms of power consumption, these devices have an outstanding efficiency, compared to GPPs they consume 100-1000x less energy[19]. Such ideal consumption values can be achieved by re-

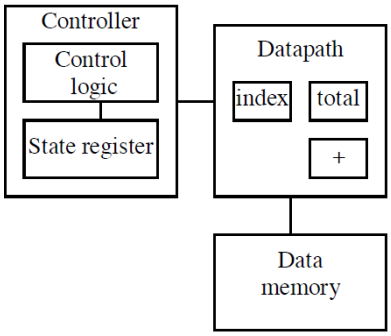
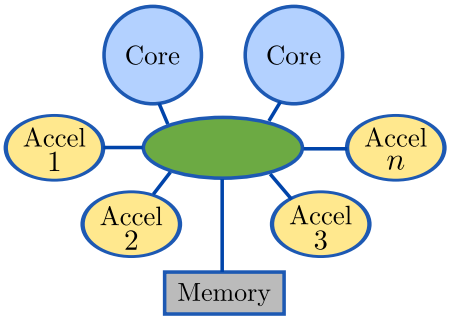


Figure 3.5: Block diagram of a simple ASIC concept [78].

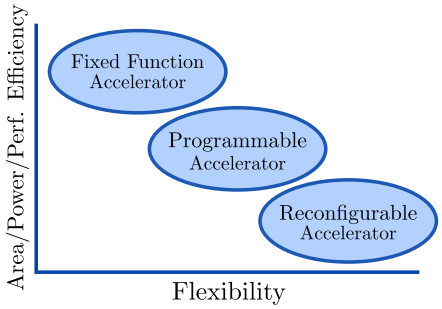
ducing flexibility and unused units. Figure 3.5 depicts a simple ASIC concept, where the control logic is reduced to a fixed state machine, and the data path only contains the required functional units for the computation.

3.2.1.1 Accelerator-Rich Architectures

Using accelerators to increase a system's performance is not a new idea. Due to the last decade's trend of more and more power-constrained and battery operated devices, the exploration of using accelerators to increase a systems trade-off between performance and power has begun [36]. Accelerators in general are developed to offer high performance while using a defined budget of energy. These components are dedicated hardware designed to accelerate a specific task, and, therefore, an ASIC. So, compared to a GPP, accelerators are able to increase the efficiency up to 500x as well as the computational performance [19]. Figure 3.6 (a) depicts the concept of an ARA, which shares the same fundamental idea of a multi-core processor system, described in Section 3.2.1. Only instead



3.6 (a): Accelerator-rich architecture concept.



3.6 (b): Trade-off between energy and flexibility for accelerators.

Figure 3.6: Accelerators in SoC design [36].

of using multiple GPP, the system is equipped with various specific accelerators, which are supporting a lightweight small programmable processing unit. Therefore, such an architecture combines the flexibility of a GPP with the performance and power efficiency of an ASIC for the specific tasks for which the accelerators were designed.

Accelerators can be designed for various application domains. Different basic categories of accelerators can be chosen for a specific design. *Fixed-function*, *programmable* as well as *configurable* accelerators are possible subdivisions. Figure 3.6 (b) shows that there exists a trade-off between energy efficiency and flexibility for accelerators, in the same fashion as discussed in Section 3.2 for processor architectures in general [36].

Compared to a processor, an accelerator traditionally tends to behave more like a producer consumer model [36], which is a big architectural implication and is important to keep in mind when ARA designs are discussed. Furthermore, it influences the communication behavior on such an architecture, as allowing a seamless integration of accelerators into the system is a challenge for hardware and software architects.

3.2.2 Overview of Recent WSN Processors

Figure 3.7 gives an overview of low-power processors developed in past decade. At this point it has to be mentioned that this plot is not a comparison. As it can be seen in the legend, the results were published using different power units, and, therefore, are not comparable.

Especially in the last two years, an increase of low power systems using accelerators, to achieve the required low power consumption, could be observed. Systems using single accelerators for improveing the algorithm performance [59, 70], or various units to compute the whole application hardware-accelerated [47, 27] were proposed. Furthermore, no-battery systems [87] as well as accelerator-based systems [22], were developed and have shown that the usage of accelerators is currently a trend in low-power processor applications.

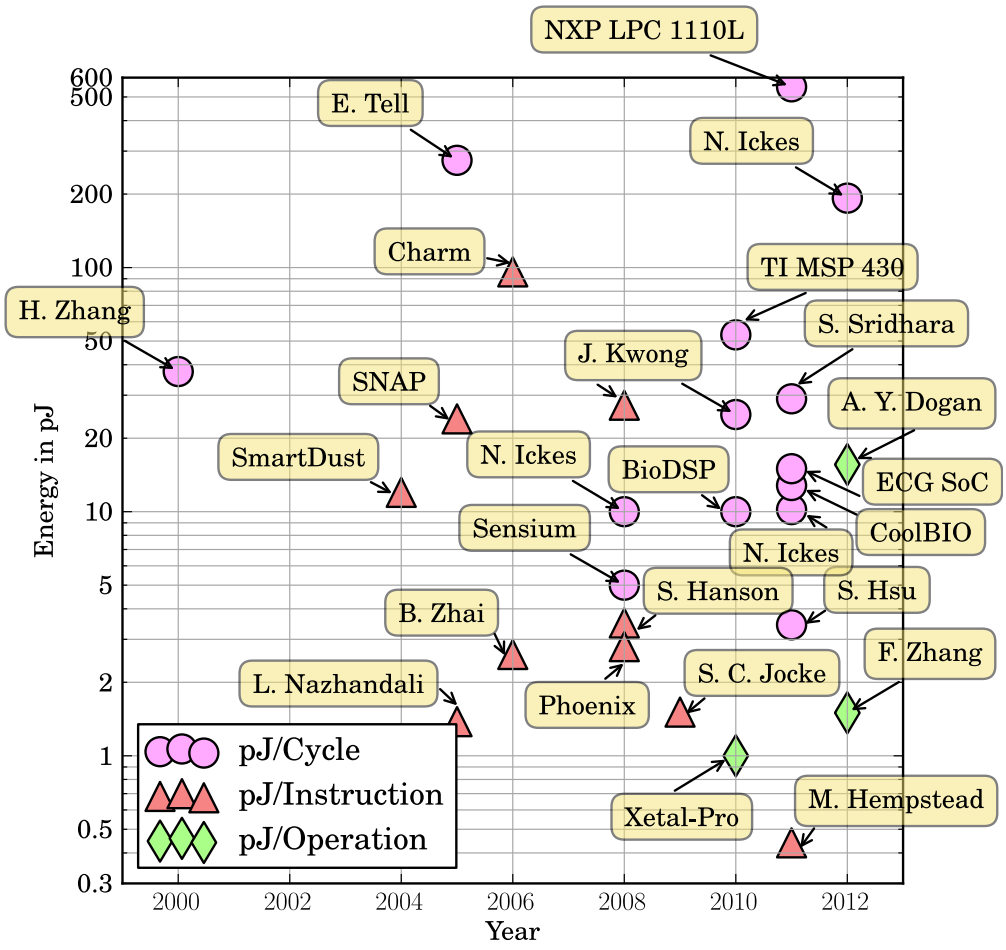


Figure 3.7: Ultra low-power processors for sensor networks in the last decade.

The corresponding data and a short design overview for each system plotted in Figure 3.7 can be found in the Appendix B.

3.2.3 Power Management in Ultra-Low-Power Processors

The power management unit is a very important component of a low-power device, in order to meet the power consumption requirements. To keep the system's consumption beneath the limited energy budget, it manages the idle and active states of all units. A fine-grained PM allows a minimization of dynamic, as well as, static power consumption [23].

Basically, the unit controls the low-power features of each component in the system. Therefore, knowledge of each component's execution state must be gathered. This information enables the PM to manage the clock and power gating or more specific features like DVS according to the systems required computational performance.

From an architectural perspective, a subdivision in *centralized* and *decentralized* power management can be made. A centralized system is a separate

component, which collects the necessary information, and decides which components have to be supplied. On the other hand, a decentralized system allows a component to dynamically decide on its own, which power state is necessary to compute the current task.

Various different approaches have been published over last years, from data-driven [73], and adaptive [1] approaches to learning-based [85] algorithms. System level communication plays an important role when managing a SoC, because be PM has to make sure that all communicating components are active before data is sent. Therefore Lahiri et al. [50] proposed a communication-based power management architecture, due to following reasons:

- Power management functionality implemented in the communication architecture would eliminate the need of a seperate PM unit, and minimize the hardware overhead.
- The communication architecture can gather the necessary information to make decisions and is able to deliver it to the related components.
- The communication infrastructure schedules communication between the units and, therefore, influences timing and execution states of system components.

3.3 On-Chip Communication Architectures in Wireless Sensor Nodes

The power consumed by on-chip interconnects is generally not the main focus of a system architect, nevertheless, interconnects are responsible for a significant fraction of the system's overall power consumption [62]. Researchers are concerned, that the structure of interconnects will be a major bottleneck in SoC architectures, due to timing delays and energy consumption [25, 62, 71]. While interconnects are required in order to allow communication in general,

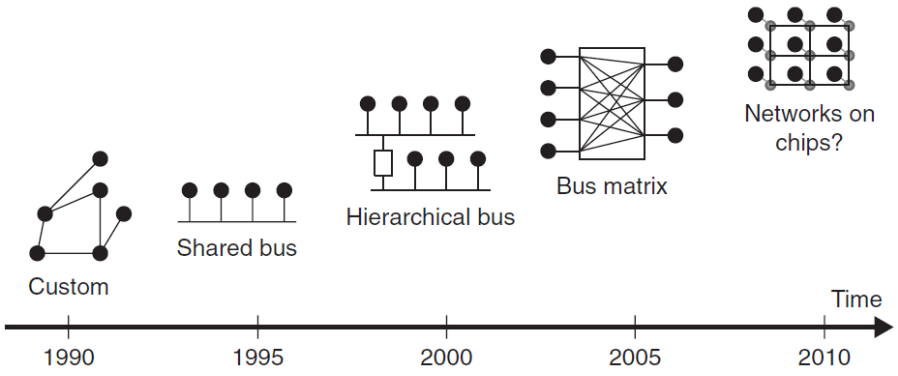


Figure 3.8: Evolution of on-chip communication architectures [60].

this section will focus on architectures, used for on-chip communication. This section will discuss state-of-the-art communication architectures, and the necessary energy for communication, as well as the suitability for low-power sensor nodes.

3.3.1 Communication Architectures

Figure 3.8 depicts the evolution of communication architectures, various architectures will be discussed in the following sections.

3.3.1.1 Custom Communication Architectures

These architectures include customized communication topologies and protocols needed in order to meet the desired performance and power goals. Such customized approaches are typically adapted to the application-specific needs of a system and, therefore, only suitable for a specific design [60].

3.3.1.2 Bus-Based Communication Architectures

Buses are a commonly-used for communication in digital designs, due to their simplicity and efficiency. The bus itself provides a shared communication channel between the components [60]. These systems can be subdivided into various categories besides their architecture, either by their physical implementation

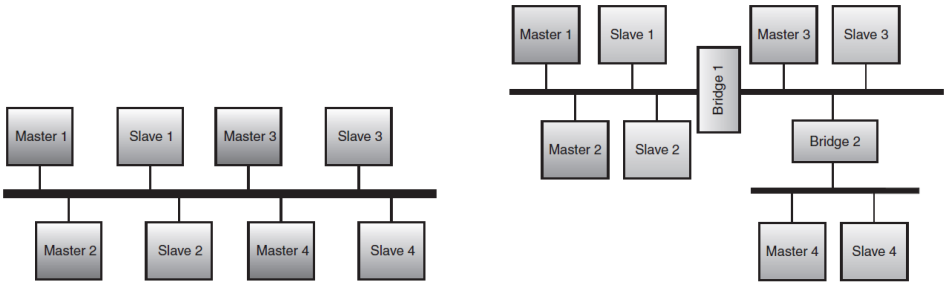
(*serial* and *parallel*) or the presence of the clock signal within control signals (*synchronous* and *asynchronous*). The signals transferred on a bus are typically classified into the following three categories:

- **Address Signals**
- **Data Signals**
- **Control Signals**

The part of the bus which transfers the address signals is commonly named the address bus, the naming scheme is the same for the other two signal categories. While communication in a synchronous bus system is synchronized via the clock signal, an asynchronous bus generally implements a handshake protocol using request (REQ) and acknowledgment (ACK) to coordinate data transfer.

Bus-based communication architectures have to be able to decode the addresses in order to select the correct destination component. Also in case of a multi-master bus system an *arbitration* mechanism is necessary to handle concurrent write requests.

Monolithic Buses are the simplest implementation scheme, a single shared channel is used to connect all components within a system, as depicted in Figure 3.9 (a). Such a communication architecture is suitable for small SoCs, containing a few units. A single bus system is not appropriate to handle communication in large systems [60]. From a power consumption viewpoint, a monolithic



3.9 (a): Simple Bus. 3.9 (b): Hierachical Bus.

Figure 3.9: Basic bus topologies [60].

bus is often inefficient for two main reasons [62]:

- For each transfer on a single shared bus the whole load capacitance has to be driven.
- The single communication channel allows only one pair to communicate, while all other units are stalled. High operating frequencies are, therefore, necessary to achieve the required data rates.

To address these problems, various extensions to the monolithic bus have been developed.

Bus Splitting or *bus isolation* is a commonly used technique to reduce the consumed power of a bus-based communication architecture. Compared to a

classic monolithic bus, energy savings in the range of 16%-50%, depending on the bus traffic, can be achieved [26]. The idea behind bus splitting is to section the single bus into multiple bus segments. Dual port drivers are used to connect the buses, retaining the functionality single shared bus [62].

Hierarchical Buses , as depicted in Figure 3.9 (b), are a further extension to the split bus scheme. A bridge component is used to enable the communication between the different bus segments. This unit can be quite complex and handle several tasks like frequency conversion and data buffering [60]. Therefore, the architecture allows simultaneous communication within each bus segment. Furthermore, a hierarchical bus can handle different bus frequencies, enabling a separation of high-traffic components, like a processor, from low-traffic peripheral components.

Bus Matrix or *bus crossbar* is an architecture designed for extensive data transfer parallelism in high performance systems. The scheme is based on point-to-point connections between the components, more precisely, in a *full crossbar bus* each master has direct connections to each slave in the system. Figure 3.10 shows such a communication architecture. A *partial crossbar bus* is a reduced

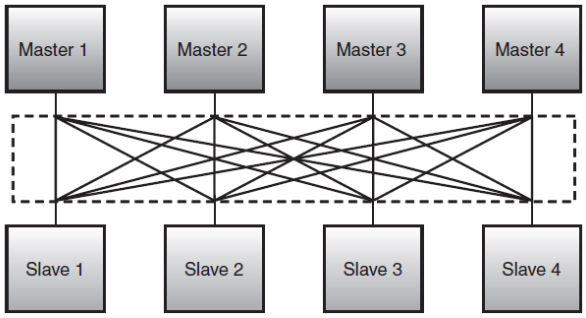


Figure 3.10: Full crossbar [60].

implementation of this topology. It uses less point-to-point connections to reduce the necessary area as well as the power consumption. So, components which do not communicate in the system and cause redundant connections are not connected. Furthermore, partial crossbars might include hybrid solutions of shared buses and point-to-point connections [60].

3.3.1.3 Direct Memory Access-Based Communication

Direct memory access (DMA) is a very common and important communication technique. It can increase the transfer rate and improves the processing unit efficiency in embedded systems [54]. The DMA approach enables communication via shared memory, where each component can read and write and therefore gather data from other units.

Figure 3.11 depicts a DMA-based communication architecture. In this configuration, a separate component, called a *DMA controller* (DMAC), handles the address and access management of the data memory. Such a component relieves a processing component from the burden of memory management. Furthermore, it allows other components, e.g. peripherals, access to the memory.

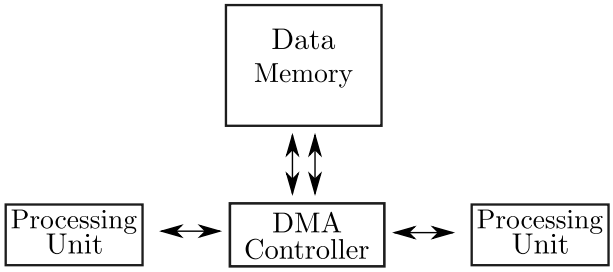


Figure 3.11: Typical DMA configuration.

A DMA can also be implemented in a distributed fashion, i.e. in a multi-master system without a controller. Each component has its own address generation and memory interface. With an increasing number of components, this can lead to very complex memory arbitration schemes.

3.3.1.4 Router-Based Communication

Using a router-based approach, instead of a classical shared communication channel, could deliver similar energy-related advantages as known from a segmented bus. Isolating all components from each other can lead to reduced load capacitance during data transfers. Furthermore, a router-based approach allows increasing data parallelism, which results in an increasing throughput [62].

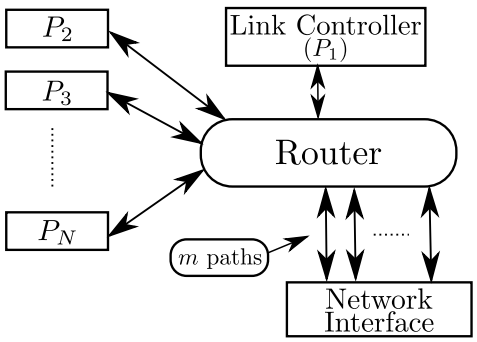


Figure 3.12: Router-based communication architecture [52].

Figure 3.12 depicts a router-based communication architecture, implemented on board-level in a sensor node platform. The energy benefits of a router-based approach increase with the number of components in the system [52].

3.3.1.5 Network-on-Chip

A Network-on-Chip (NoC) is the approach of scaling down a classic network concept to embed it on a SoC. Network components like routers and network interfaces are, therefore, necessary. These architectures were designed to handle communication of hundreds of processing engines. An implementation of a whole network protocol stack similar to the ISO/OSI model suits the architecture for such large numbers of components. While a NoC is a very promising architecture for large SoCs, it is less attractive for systems with less aggressive requirements [60].

Especially in WSN the systems used in the sensor nodes tend be small and efficient, even ARA employed in the processing units will not lead to such high numbers of components. Therefore such an architecture would cause too much overhead in such small-scale systems.

3.3.2 State-of-the-Art Communication Systems

In the following section various off-the-shelf communication architectures are discussed.

The Advanced Microcontroller Bus Architecture (AMBA) is nowadays one of the leading communications standards. While the three main bus architectures were already defined in AMBA 2.0, the newer version AMBA 3 added additional interface protocols. Furthermore, the current AMBA 4 extended the specification and added the *Advanced eXtensible Interface* (AXI) [2].
AMBA uses three different bus standards [60]:

- **Advanced High Performance Bus** (AHB), suited for high performance communication, necessary for on-chip memories and processors.
- **Advanced System Bus** (ASB), an AHB alternative with a reduced protocol feature set.
- **Advanced Peripheral Bus** (APB), a low-power optimized bus suitable for low bandwidth components.

The newest interface protocol AXI4 includes a streaming interface optimized for point-to-point connection, specialized for data-intensive multimedia applications [2]. AMBA supports hierarchical bus architectures as well as bus splitting and bus matrices for high bandwidth applications [60].

IBM CoreConnect is a synchronous bus standard, very similar to AMBA, which is also commonly used in SoC design [60]. The CoreConnects architecture also includes three busses:

- **Processor Local Bus** (PLB), a high performace bus, very similar to the AHB.
- **On-Chip Peripheral Bus** (OPB), which includes a more advanced feature set then the AMBA APB.

- **Device Control register** (DCR) bus, which is designed to reduce the memory mapped configuration registers and replace it with a separate light-weight control bus.

CoreConnect also supports hierarchical architectures and bus splitting.

Sonics SMART Interconnect & STMicroelectronics STBus are two additional commercial bus-based systems including different types of bus standards suitable for applications similar to the systems mentioned above. For futher details on these bus implementations, the interested reader is referred to the literature [2, 60].

OpenCores Wishbone is an open-source bus-based communication architecture, which defines interconnection schemes to allow quick and easy *intellectual property* (IP) integration [2]. Wishbone is highly configurable and, therefore, allows application-specific customization. It supports all common architecture types like shared bus, point-to-point, crossbars as well as a data-flow topology for sequential processing of a data stream [60].

3.3.2.1 Socket-Based Interface Standards

Such standards describe the interface connecting a component with a communication architecture. As Figure 3.13 depicts, a socket-based interface standard allows implementation of units without specifying the particular architecture used to communicate. For this reason using these standards improves the reusability of designed IPs [60].

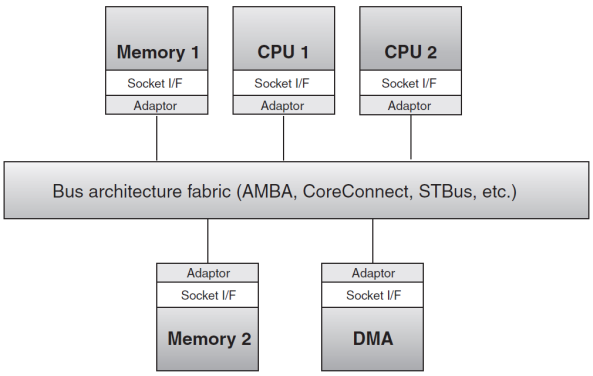


Figure 3.13: A system using socket-based interfaces [60].

Three interface standards exist, which can enable a "plug-and-play" SoC design:

- **Open Core Protocol** (OCP)
- **VISA Virtual Component Interface** (VCI)
- **Philips Device Transaction Level Protocol** (DTL)

The OCP is strongly supported by the industry and is seen as the universal complete socket standard [2]. AMBA, CoreConnect, STBus are perfectly suitable using the open core protocol. Furthermore, the Sonics SMART interconnect has native support of the OCP 2.0 standard [60].

3.3.3 Power Consumption Through Communication

In low-power systems it is necessary to minimize all available sources of consumption, therefore this section will take a close look at the required power for communication. Figure 3.14 depicts the power consumed in a hierarchical AMBA implementation. In this application the total consumed power used for communication was 12 mW [60]. Furthermore, it shows that the bus wires con-

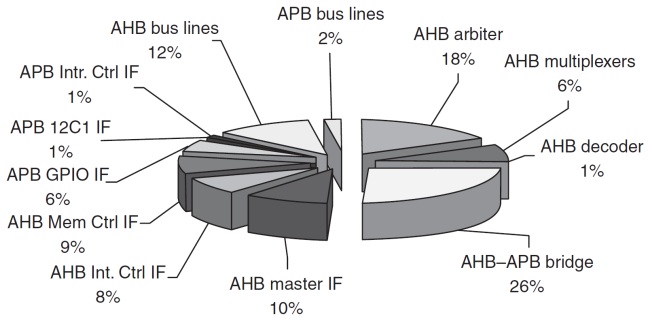


Figure 3.14: Power consumption breakdown of a simple bus-based communication architecture [49].

sumed 14% of the whole budget, which is quite a small fraction compared to the power consumed by all other necessary units. Bus segmentation applied in this system was able to save 70% of the power consumed by the bus lines [49]. In such classical bus-based architectures the main part of the power is used for the communication overhead, more precisely, interfaces, arbiters and bridges, which are necessary to ensure functionality in a shared bus system.

For other communication architectures which use even more resources, like a DMA-based communication architecture, a higher total power consumption can be expected. Especially due to the fact that memories are main contributors to a processing unit's consumption.

3.3.4 Communication Architecture Requirements for ARAs

Connecting various components in an effective fashion while staying within a limited energy budget is a hard challenge for a communication system applied in an ARA. Furthermore, the accelerators used are typically not implemented in way to allow light-weight and efficient communication. More precisely, accelerators tend to use large memories and communicate via complex DMA schedules [53].

- **Support for Differing Voltages and Frequencies:** On architectures containing various different processing units, it is worthwhile that these

components might implement specific low-power features. Therefore, the communication architecture should be suited for handling data transfers between differing voltage islands and clock frequencies.

- **Close Connection to PM Unit:** the communication system should be tightly connected to the PM unit, to prevent transfers to deactivated components as well as to allow efficient power management as discussed in Section 3.2.3
- **Small Communication Overhead:** Minimizing the communication overhead is important to allow a power-efficient implementation.

3.4 A Low-Power WSN Processing System: ECGSoC

This section will present the low-power sensor node processing system, which is the foundation of this master thesis. Information of the chip is mainly derived from [41], the system's technical documentation [43] and the work with the system itself.

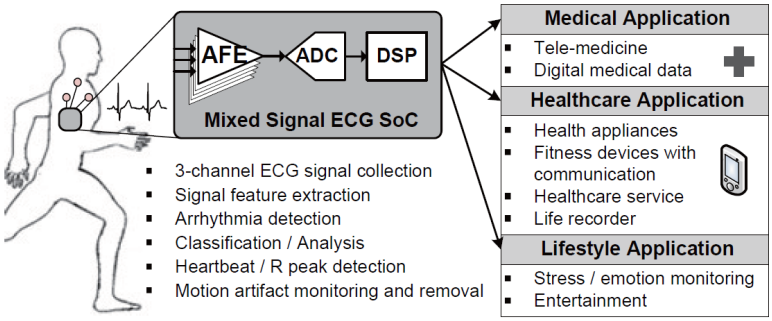


Figure 3.15: Mixed-signal ECGSoC and typical applications [41].

Figure 3.15 depicts a feature and application overview of the mixed-signal SoC designed for portable ECG monitoring applications. The design enables configurable functionality as well as a low power consumption.

The analog front end allows high quality extraction of 3-channel ECG signals as well as single channel impedance measurement. The digital back-end provides the system's configurability and ECG processing functionality using a custom DSP-like ASIP implementation.

The focus of this thesis is the optimization of the system's digital back-end and, therefore, the AFE will not be discussed in further detail.

3.4.1 Implemented Algorithms

Three different real-time ECG monitoring applications are implemented on the ECGSoC:

- **Data Collection:** This application uses the AFE to simultaneously record the signal provided by the selected channels.

- **Beat Detection:** This mode allows QRS-complex detection executed with the help of a derivative-based algorithm [42] or a band-power-based low-power beat detection algorithm [84].
- **Accurate R-peak Search:** This operation mode allows a highly accurate R-peak detection, including a motion artifact removal stage to increase the signal reliability as well as the automatic analysis performance.

Figure 3.16 depicts the various algorithms including their execution stage, within the digital back-end. The data received from the AFE, is separated by a data demux unit in hardware. In case of the high accurate extraction mode, either PCA or an adaptive LMS is used to remove the motion artifacts. The ECG channel used for the adaptive filter can be configured. For robust feature extraction a

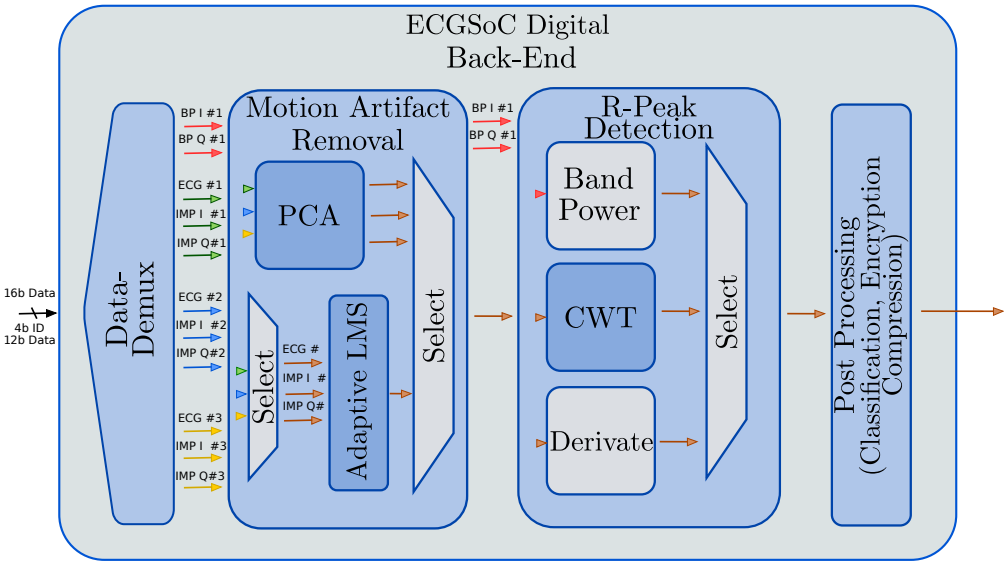


Figure 3.16: Algorithm overview.

CWT-based algorithm [65] processes the filtered signals.

For low-power beat detection based on the signal's band power, the AFE provides the necessary signal. Furthermore, the automatic analysis is directly performed without any pre-processing.

The derivative-based algorithm can be executed either on a pre-filtered signal or the captured original ECG.

In the thesis, the focus of attention is the highly-accurate R-peak detection algorithms. Due to the necessary filtering and computation-extensive feature extraction this operation mode requires more energy.

3.4.2 Architecture Overview

The ECGSoC is implemented in a 0.18 μm process and operates at a clock frequency of 1 MHz. Either the integrated ring oscillator of an external clock source can be used to provide the system's clock signal. The nominal supply

voltage of the process is 1.8V, but the system is fully functional using a down-scaled voltage of 1.2 V. Three memories are used within the digital back-end, the data memory (32 kbyte), the program memory (5 kbyte) and the coefficient memory (4 kbyte). The whole digital part of the mixed-signal SoC accounts for 6.3 mm^2 chip area.

Clock gating is used to reduce the dynamic power consumption of the system, therefore five different clock domains are implemented. As Figure 3.17 shows, the system also includes various interfaces, a host SPI interface, allowing access to the system memories, a JTAG debugging interface as well as an SPI interface to the AFE. A pre-processing unit is implemented to allow pre-processing of the captured signals, before storing the data in the memory. A DMA controller is

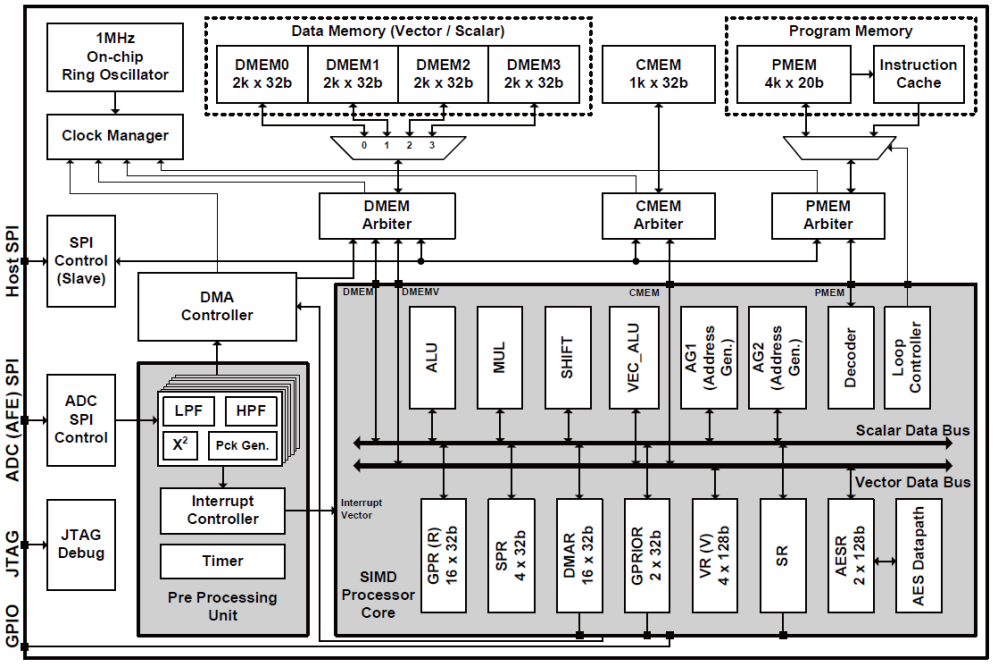


Figure 3.17: Blockdiagram of the digital back-end [41].

responsible for controlling the data memory accesses by the preprocessing unit and the processor core.

3.4.3 Application-Specific Instruction-Set Processor

The customized DSP-like ASIP includes a 4x32-bit SIMD vector data-path as well as a 32-bit scalar data-path. The data memory is, therefore, split into four equal-sized memory banks, allowing simultaneous access to enable one-cycle data vector fetching. Furthermore, the program memory is extended with a 128-word loop instruction cache to reduce the number of memory accesses within a loop execution. Various other features of the processor are implemented to allow effective ECG processing:

- A custom 20-bit instruction set
- A parallel single cycle execution of memory loads, ALU execution
- A 32-bit multiplier
- Two address generation units
- Unaligned vector memory access
- A register-mapped DMA-controller configuration
- A 9-bit interrupt vector

3.4.4 Application-Specific Accelerators

Dedicated hardware is used to accelerate pre-processing and the encryption of the ECG signals. For each ECG channel, the system implements a dedicated FIR and a IIR filter to enable accelerated low-pass and high-pass filtering within the pre-precessing unit. Additionally to the filters, a squaring unit is implemented for pre-processing.

To allow secure wireless transfer of the monitored signal, an AES-128 encryption unit is attached to the ASIP. Configuration, as well as, communication to the accelerators is register-mapped.

3.4.5 Communication Architecture

On the ECGSoC the communication between the pre-processing unit and the processor is implemented in a DMA-based fashion. This allows the AFE interface to continuously write the captured data into the memory without the need to interrupt or even enable the processor. When the address range of the data

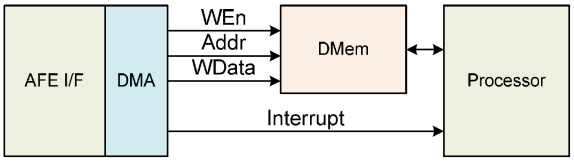


Figure 3.18: Block diagram of the ECGSoCs DMA-based communication [43].

memory, used to buffer the input signals, is full, the AFE interface wakes the processor with the help of an interrupt.

The size of each input buffer range can be defined per input channel via the DMA controllers configuration registers. The DMA controller handles the arbitration between the available bus masters: the AFE interface, the processor and the host SPI interface. Using multiple masters on the shared memory necessitates the implementation of an address generation unit for each master. Furthermore, the controller has to avoid simultaneous access, by a prioritizing scheme. On the ECGSoC, the avoidance of collisions is solved by halting the clock signal of the unit with the lower priority, which can cause delays.

3.5 Previous Work on ECG-Monitoring Systems

Besides the ECGSoC, numerous solutions were published addressing the challenge of portable low-power ECG monitoring.

3.5.1 ASIC Implementations

- [84] presents an analog signal processor specialized for ECG monitoring. The system uses and adaptive analog-to-digital conversion scheme to sample and process the signal according to its frequency content.

3.5.2 Configurable Implementations

- In [27], a hybrid biomedical signal processor is introduced. Using multiple functional units, specialized for biomedical algorithms, encryption or general-purpose use, optimizes the system in terms of flexibility and power dissipation.

- A mixed-signal SoC deploying a sub-threshold microcontroller is described in [37]. Using various analog and digital sub-threshold circuits minimizes the systems power consumption.

- [47] presents an energy-efficient signal processing platform specialized for biomedical applications like EEG and ECG. The system uses a lightweight microcontroller equipped with multiple accelerators to allow effective signal processing. To ensure flexibility, the platform uses DMA-based communication.

- In [42] a custom ECG signal processor is introduced, specialized for ambulatory arrhythmia monitoring. The processor uses three heterogeneous processing elements for efficient filtering, classification, compression and encryption. To achieve a low-power consumption clock gating as well as voltage scaling is applied.

With regards to communication architectures it has to noticed, that all systems for ECG monitoring which use accelerators to reduce their power consumption are using either full custom communication architectures or custom DMA-based systems.

Chapter 4

Design of the Dataflow-Oriented ECGSoC

4.1 Introduction

This chapter presents the fundamental components of the design process for the dataflow-oriented ECGSoC. A redesign of the ECGSoC described in Section 3.4, focusing on minimization of the system's energy consumption using conceptional modifications of the processing unit's architecture as well as the used communication infrastructure.

First, the environment of the ECGSoC processing unit is discussed in more detail than in Section 3.4, which served as an overview. Second, the general idea of the streaming-based architecture is presented. Furthermore, the changes required bu applying the new ECGSoC architecture are outlined. Requirements and a concept for a generalized component architecture are examined.

4.2 Digital Back-End Environment

The mixed-signal ECGSoC is separated into two parts, similarly to the typical sensor node structure discussed in Section 2.5. The AFE handles signal readout and the analog-to-digital conversion separately from the digital back-end.

Therefore the analog part includes low-noise and a low-power instrumentation amplifier as well as several analog filters for each channel. A 12-bit successive approximation ADC, supporting an adaptive sampling scheme allows ECG data compression, prepares the data for the digital processing.

The digital back-end receives the data from the AFE via a separate SPI interconnect. The data is transferred using 16-bit packages, consisting of 12-bits data and 4-bits storing the channel identification.

After the post-processing stage, the data which should be transferred is stored in the data memory. A host processor can read out the resulting information via the SPI interface. The processed data is collected by a MSP430 on the ECGSoC evaluation board [43].

4.3 Streaming-Based Architecture Concept

Various processing systems applying the ARA have shown an impressive improvement of the energy flexibility trade-off when deployed for a defined set of applications [22, 47]. The idea of a streaming-based system-level architecture is depicted in Figure 4.1, including various accelerators supporting a light-weight GPP. The fundamental concept separates the system's communication architecture into two communication channels.

The control channel provides an infrastructure to handle irregular events such as configuration during the boot phase and furthermore enables application-specific reconfiguration of the used accelerators at the execution stage.

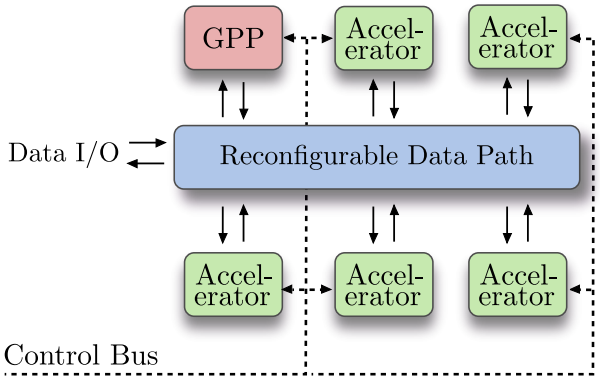


Figure 4.1: Basic concept of a dataflow-oriented hardware-accelerated processing architecture.

The separated data channel allows efficient sequential processing of the captured data. The reconfigurable data path provides a flexible communication structure enabling an application-specific algorithm mapping. More precisely, the various computation steps of an application can be mapped to the appropriate processing element, allowing effective and power-efficient processing. Furthermore, the reconfigurable data path makes the usage of an additional address bus within the data channel redundant, which can improve the efficiency in terms of power consumption and performance.

For algorithms which allow completely sequential processing it enables on-the-fly computation using various processing elements in the corresponding order. Therefore the architecture reduces the necessary data caching between the various execution steps.

As described in Section 3.3.4 using various IPs within a system might necessitate the need for communication between different voltage islands and/or clock regions. Due to the possible delays between components with different operating frequencies using asynchronous communication within the data channel represents a valid option. Furthermore, a suitable foundation to efficiently manage used low power techniques of an accelerator is provide by using request and acknowledge based protocols. Therefore, a suitable power management unit can control the accelerators based on the REQ/ACK signals provided by the

communication infrastructure.

Reconfigurable interconnects are also used in systems for reconfigurable computing [88]. This approach shares the same basic idea with the proposed system. Heterogeneous reconfigurable architectures have been developed to map algorithms directly to dedicated hardware. In contrast to the proposed architecture, the reconfigurable computing approach uses processing elements on a much smaller scale. The proposed system uses large-scale accelerators able to execute specific algorithms, while reconfigurable processors use dedicated hardware implementing mathematical operations.

4.3.1 Streaming-Based Connection

To allow a reconfigurable connection between various different components, a common denominator has to be found, which defines the necessary communication between two units. Furthermore enabling power management based on the information provided by the communication channel necessitates a common standard of control signals to ensure clearness about the current execution state. Figure 4.2 depicts the minimum of information required to enable direct

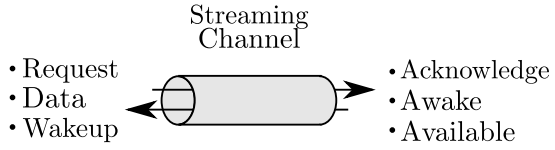


Figure 4.2: Information transferred within a streaming channel.

streaming between two components and ensure the control possibility of low power techniques. The request and acknowledge pair controls the transferred data, an available flag provides clearness about the resources of the component and an awake and wake-up pair allows control of low-power features like power or clock gating.

4.3.2 Reconfigurable Datapath

The reconfigurable datapath is basically a switchable connection matrix, offering point-to-point connections for each component. This system is very similar to a bus matrix applied in bus-based communication architectures, described in Section 3.3.1.2. This datapath can be configured to connect the systems components in a variable order. Therefore, the system allows changing the execution order of accelerated algorithms, which enables flexibility in the executed application. Furthermore, the application can be adapted via software.

4.4 Architectural Changes Applied to the ECGSoC

Using a dataflow-oriented architecture to enhance the ECGSoC in terms of power consumption leads to various changes within the digital back-end. This section will discuss the impact on the design, caused by the architectural change. As already mentioned in Section 3.4.1, this thesis focuses on the highly-accurate R-peak search operation mode and, therefore, will only take the influence of the architectural redesign according to that application into account.

4.4.1 Dataflow Within the ECGSoC

Before discussing the application of the datafow-oriented architecture, it is necessary to take a close look at the existing dataflow within the ECGSoC.

Figure 4.3 visualizes the information flow of the accurate R-peak detection application. This application is subdivided into two use cases, depending on the algorithm used for motion artifact removal. The data received from the AFE is redirected into the data memory. This task is executed by the AFE interface, which decodes the channel identification and stores it into the appropriate address range.

When the input buffer is filled, the ASIP starts to process the data. First, dependent on the use case scenario, a motion artifact removal algorithm, either an adaptive LMS or a PCA, processes the input signals stored in the memory and writes the resulting cleaned ECG signal back into the storage.

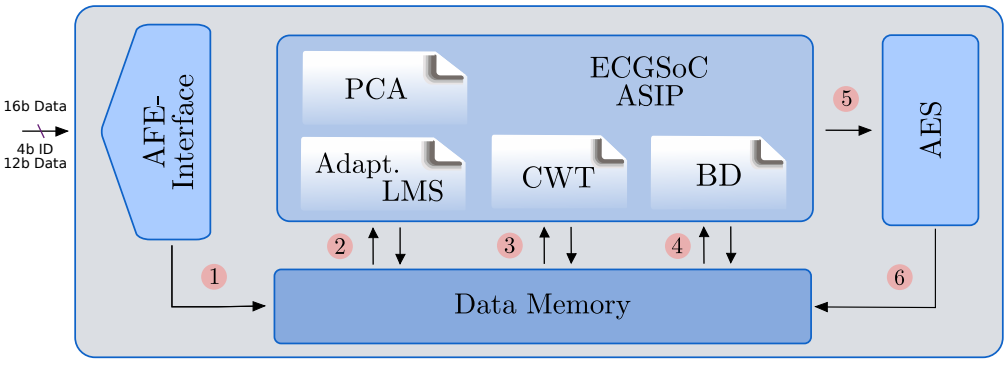


Figure 4.3: Visualization of the dataflow within the ECGSoC.

The CWT used for feature extraction is executed and uses the clean ECG signal as an input. After processing, the algorithm stores its results in the memory. In the final processing step the ASIP executes the beat detection (BD) algorithm, manipulating the information stored in the data memory again.

Afterwards, the post-processing steps are executed. The ASIP transfers the information to the encryption accelerator and again stores the result into the data memory, where a host processor can read out the processed ECG information via the SPI interface.

4.4.2 Applying the Dataflow-Oriented Concept

Figure 4.4 depicts the concept applied to the ECGSoCs architecture. The AFE interface is used to decode the channel identification to allow correct allocation within the datapath. Furthermore, the implemented pre-processing unit within the interface will still enable channel-specific filtering.

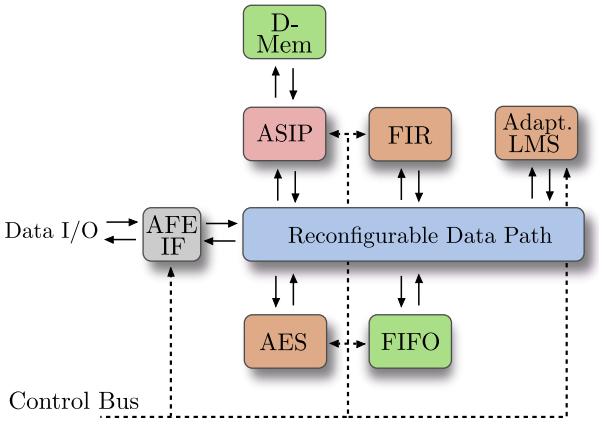


Figure 4.4: Basic Concept of a dataflow-oriented approch applied to the ECG-SoC.

To accelerate the feature extraction, a reconfigurable FIR filter is attached to the system. The CWT algorithm has a FIR structure itself and is, therefore, perfectly suited for the dedicated hardware unit. Furthermore, a FIR accelerator fits into the data-streaming concept per definition.

The adaptive LMS algorithm is also per definition applicable within a streaming processing system. Therefore, the architecture is extended with a 4^{th} order LMS accelerator.

Due to the block-based nature of the PCA algorithm an accelerator would necessitate large implemented memories for efficient performance. Therefore, this motion artifact removal algorithm will be executed by the ASIP.

The evaluation of the ECGSoC has shown that the data memory is the main power consumer within the digital back-end, this will be discussed further in Section 6.2. To unburden the data memory from continuous write accesses during data collection, the system is equipped with a separate input buffer. These buffers enable storing the gathered ECG signal and will wake up the ASIP to process the information when they are filled. To apply these buffers within a streaming based computation flow, a First-In, First-Out (FIFO) structure offers a valid option. Furthermore, a separate input buffer can reduce the system's power consumption, due to the fact that a smaller memory will consume less internal and leakage power.

An additional advantage of a FIFO within the reconfigurable data-path is that the system architect can decide at which execution stage of the application the data will be stored.

The system is, therefore, able to use the accelerators and execute various parts of the application without the presence of a processor or data memory.

4.4.3 Proposed Dataflow Within the Dataflow-Oriented ECG-SoC Architecture

Using the dataflow-oriented architecture has a strong impact on the processing flow within the system. Figure 4.5 depicts the proposed dataflow within the new architecture for the two different use cases of the high accurate R-peak detection application.

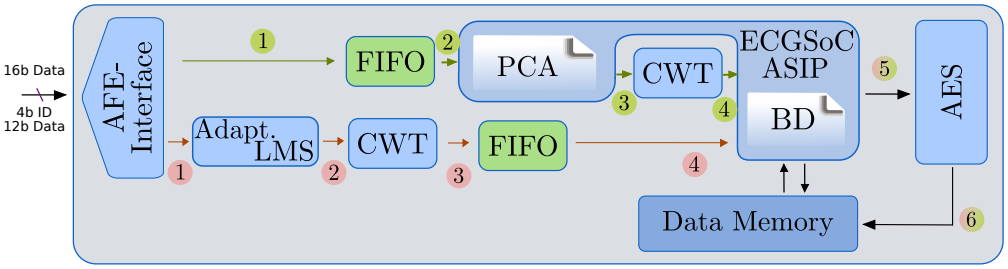


Figure 4.5: Proposed dataflow within the new architecture.

The details for both use cases will be discussed in the following sections.

4.4.3.1 Adaptive LMS for Motion Artifact Removal

The red arrows and numbers in Figure 4.5 indicate the adaptive LMS use case. The AFE interface separates the ECG signal and the data from the impedance measurement and sends the data directly to the adaptive LMS accelerator. The motion artifact removal algorithm processes the data sample per sample and transfers the resulting cleaned signal to the feature extraction stage. The implemented reconfigurable FIR filter executes the CWT algorithm and sends the data to the input buffer. Allowing the accelerators to process the captured signals reduces the data amount by 50% before it reaches the FIFO. When the buffer is filled, the ASIP wakes up and executes the beat detection algorithm. After the automatic ECG analysis, the data is sent to the encryption accelerator and stored in the data memory so a host processor can gather the results. In the ideal case the system will only store the resulting encrypted information into the data memory, and during all other execution stages, the data memory is not required.

4.4.3.2 Principal Component Analysis for Motion Artifact Removal

The dataflow within this use case in Figure 4.5 is indicated by the green arrows and numbers. Due to the block-based nature of the PCA algorithm, the gathered ECG signals are sent directly to the input buffer. After collecting enough data the ASIP awakes and executes the motion artifact removal algorithm. Therefore the whole input data has to be copied into the data memory,

after which the resulting clean ECG signal is read from the memory and sent sample by sample to the FIR accelerator. The feature extraction stage executes the CWT algorithm again and transfers the data back to the ASIP. The processor uses the beat detection algorithm to compute the results and transfers the data to the AES accelerator. Concluding the ECG processing, the encrypted results are stored into the data memory.

4.5 Algorithmic Changes

This section discusses the influence of the architecture on the executed algorithms in the high accuracy R-peak detection application.

4.5.1 Motion Artifact Removal

The *PCA* algorithm itself is not going to change due to the new architecture, it still is executed by the ASIP. But it is necessary to modify the duty cycle as well as the block length of the processed signal to adopt the algorithm according to the input buffer length. In the original system this length was limited by the size of the data memory.
The *adaptive LMS* used for motion artifact removal will be implemented on dedicated hardware. Therefore, the same algorithm of the original system will be used: a 4^{th} order *normalized* LMS (NLMS).

4.5.2 Feature Extraction

The *CWT* maximizing the R-Peaks will be executed by the FIR accelerator. This configurable accelerator enables flexibility in its coefficients and, therefore, allows algorithm updating via software. The same algorithm as in the original system will be used, although the number of coefficients and the coefficients will be updated to the current best performing configuration. These values are provided by the IMEC-NL algorithm department and represent the state-of-the-art R-peak highlighting.

4.5.3 Beat Detection

The beat detection algorithm in the original system was a part of the CWT algorithm, more precisely, within the feature extraction stage intervals were marked which contain the R-peak. The beat detection itself used these marked boundaries to execute a maximum search on the cleaned ECG signal within the intervals.

Based on the on-the-fly computation executed by the new dataflow-oriented architecture, the cleaned ECG signal in not present after the feature extraction stage. This necessitates usage of another beat detection algorithm based on the resulting data of the CWT algorithm.

In order to allow streaming-based beat detection the new architecture will use the algorithm depicted in Figure 4.6. The algorithm uses an adaptive threshold to detect an occurring peak within the signal. If the signal exceeds the actual

threshold a counter starts. This counter is used to define the interval in which no higher peak is allowed to occur, otherwise the interval starts again. Because, especially within the QRS-complex of an ECG signal, the CWT algorithm causes multiple peaks next to each other. The algorithm detects the maximum value within the interval. Based on the index value, the time-stamp of the R-peak can be reconstructed.

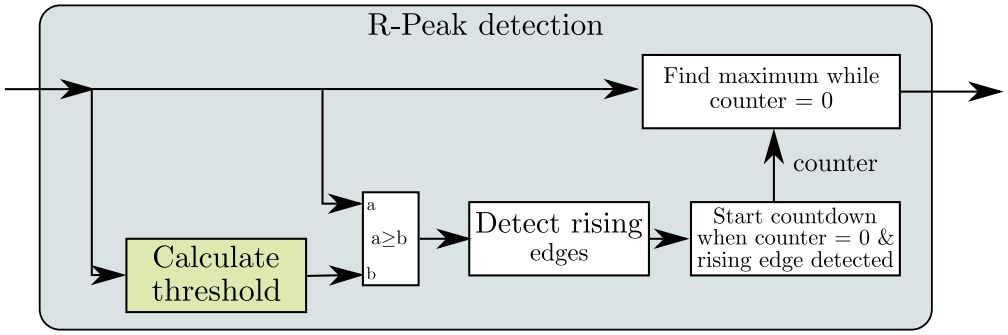


Figure 4.6: R-peak detection algorithm.

The threshold can be configured using a gain and an offset parameter. A *quasi-peak-detection* (QPD) is used to adapt the threshold during execution. Figure 4.7 shows the block diagram of the adaptive threshold calculation. Depending on the difference between the last threshold and the actual sample the QPD either uses the *attack* coefficient (α) or the *decay* coefficient (β) to adopt the threshold. These two coefficients allow a configuration of the rise and fall times

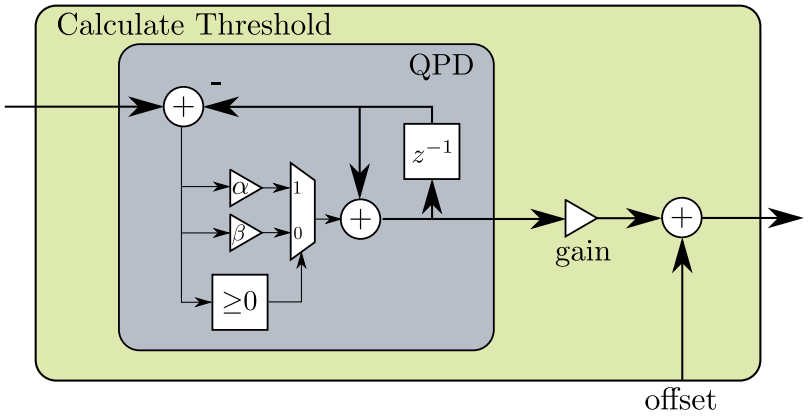


Figure 4.7: Adaptive threshold calculation.

of the threshold. Furthermore due to the influence of the difference between the signal and the threshold, this algorithm suits the beat detection in ECG signals more precisely to detect single high peaks within a signal.

One of the big advantages of this algorithm is the minimal necessary space in the data memory. Only the last threshold value has to be stored during the processor's sleep phases. Furthermore, the algorithm combined with the CWT

reached excellent performance when it was verified based on the MIT ECG database at IMEC-NL.

4.6 Accelerator Design

This section discusses the design of the various modules within the proposed system. To prevent unnecessary I/O complexity as well as sources for bottlenecks the modules should be small, with good performance and enable efficient communication [53].

4.6.1 Abstract Module Design

In order to allow a module to work within a reconfigurable data-path it is necessary to define a common denominator within the design. More precisely, functionality of a module should not influence the general communication behavior of the system. Therefore, the proposed architecture uses a common universal module architecture as depicted in Figure 4.8. Furthermore, this architecture offers a unified interface to the systems power management unit, to support various low-power techniques, like clock and power gating.

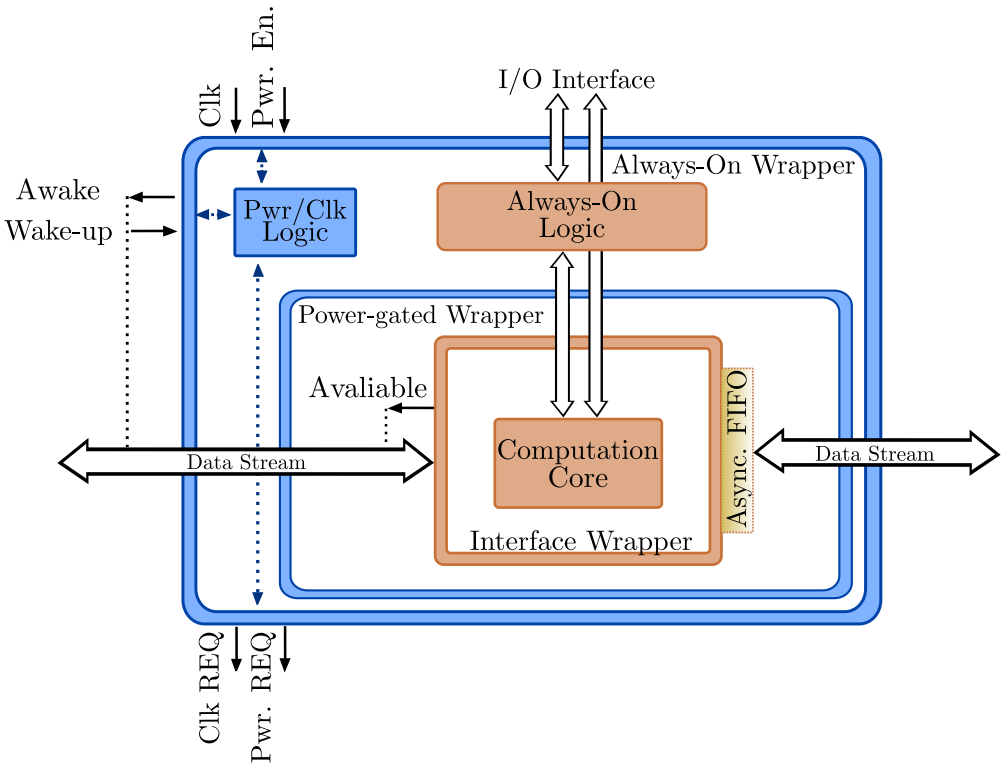


Figure 4.8: Streaming-based model design.

A module is separated into two parts: a core part, which includes the algorithm implementation and the according control logic, and a part containing the re-

quired logic necessary to store information like execution state, coefficients or previous processed data. This separation allows applying power gating to the core without losing required information to restart computation after waking the module. So all parts inside the *power-gated wrapper* cannot retain information during sleep phases.

The *interface wrapper* has the important role to ensure communication between modules. This wrapper prepares streamed data so it can be processed by the computation core. Furthermore, the wrapper is also responsible for supervising the execution states of the computational core. To prepare the interface for communication between modules using different clock frequencies, an asynchronous FIFO can be deployed.

To allow configuration, the module is equipped with an input/output interface. This interface can either be a custom implementation or a standard bus interface. Furthermore, this interface as well as the communication between the computation core and the always-on logic can vary between the modules to ensure flexibility and allow an efficient implementation.

Each module contains its own power and clock control logic. This allows a unit to optimize the usage of the built-in low-power features. Based on the wake-up information transferred via the streaming datapath, a module can request the power management unit to supply it according to the actual execution state.

An additional effect of a standardized module structure is that a big part of it can be automatically generated based on the information of the core modules. The blue-colored parts in Figure 4.8 indicate that these can be created automatically.

4.6.2 Adaptive LMS Accelerator

The adaptive NLMS algorithm used for motion artifact removal is an extension to the adaptive LMS described in Section 2.4.1.2. The extension is confined to the step size parameter μ, to ensure convergence for highly varying input signals the parameter is adjusted to the actual input signals. Therefore the μ_0 is divided by the norm of the actual input signals, as presented in Equation 4.1. This necessitates an additional division, but offers better performance for ECG signal processing.

$$\mu = \frac{\mu_0}{||\vec{x}[n]||^2} \tag{4.1}$$

Figure 4.9 depicts the structure of the 4^{th} order adaptive NLMS algorithm. In order to apply the module architecture standard, as discussed in Section 4.6.1, the delay tap lines storing the coefficients and the past input samples have to be implemented within the always-on logic. All multipliers, adders and division blocks can be part of the computation core, since they do not contain information necessary for the following computation.

4.6.3 Dedicated FIR Filter

Due to the fact that the CWT per definition has symmetric coefficients, using a folded delay line for the FIR filter implementation is a valid option. Therefore,

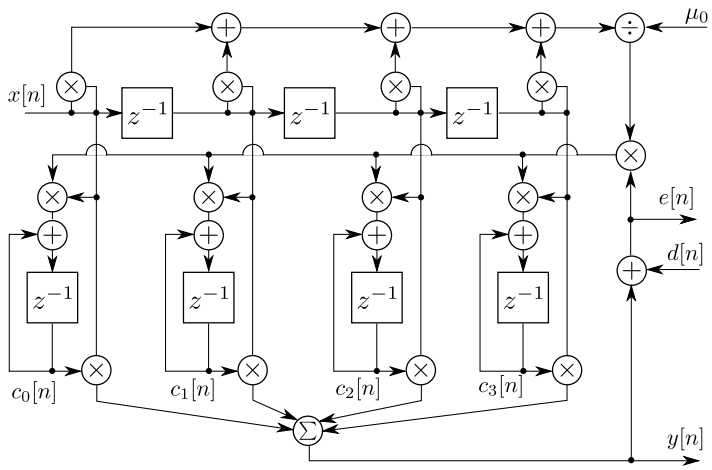


Figure 4.9: Structure of a 4^{th} order adaptive NLMS algorithm.

only 50 % of the multipliers, compared to a traditional implementation are necessary. Furthermore, to the reduced number of multipliers in the design is smaller and more power-efficient. To implement the filter according to the

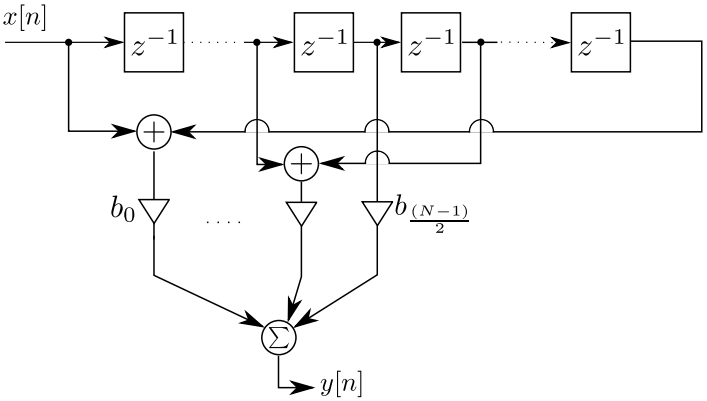


Figure 4.10: FIR structure with a folded delay line.

module standard defined in Section 4.6.1, the folded delay line, as well as the coefficients, have to be part of the always-on logic to ensure functionality. The multipliers and adders together are the power-gateable computational core of the accelerator.

This page is intentionally left blank.

Chapter 5

Implementation of the dataflow-oriented ECGSoC

5.1 Introduction

This chapter presents the implementation of the dataflow-oriented ECGSoC from an algorithmic as well as an architectural perspective. Furthermore, applied optimizations regarding power and performance are outlined as well as the used design tool flow is presented.

First, the design tool flow employed within this thesis is introduced. Second, the implementation of the accurate R-peak detection application ranging from the reference model to the specific algorithms executed at the various processing engines is discussed. Furthermore, a detailed description of various added components is provided. Concluding the chapter, an overview of the implementation procedure is given, from the high-level description down to the physical layout.

5.2 Tool Flow

Various tools are involved in the design and implementation process of the new ECGSoC architectures. From a high-level architectural description of the SoC's top-level, the processor description of the implemented ASIP, various HDL models of the attached accelerators, the application executed at the ASIP, the following architecture characteristics can be derived:

- **Execution Performance:** expresses the application execution time as well as the cycle count.
- **Physical Characteristics:** shows the required silicon area and the number of gates.
- **Consumed Power:** represents the dissipated power and energy of the system.

Figure 5.1 gives an overview of the used design tool flow. The tool flow used to implement the new ECGSoC architecture can be subdivided into three sub-flows:

- The **Synthesis flow** synthesizes the architecture using a 180nm TSMC standard cell library. After placing and routing the system, the physical characteristics can be obtained.
- The **Application performance flow** is necessary to compile, assemble and link the application to the used GPP. Simulating the architecture executing the application with the netlist simulator produces cycle-accurate performance figures.
- The **Power estimation flow** estimates the architecture's power numbers by combining the information from the synthesis and application flow.

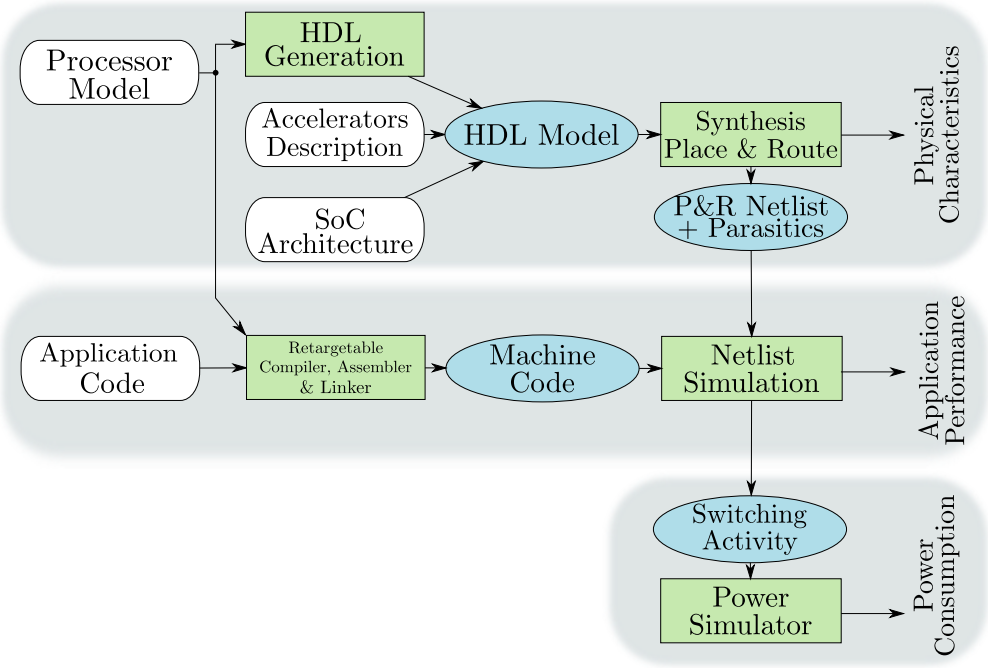


Figure 5.1: Design tool flow overview.

5.2.1 Synthesis Flow

The synthesis sub-flow is used to obtain the physical characteristics of the given architecture. The dataflow-oriented ECGSoC is separated into three parts. First, the processor model of the deployed ASIP is described using the nML language. The custom instruction as well as the user-defined primitives for the processor implementation are included in this model. Using the Target Go HDL generator [74], a Verilog processor model is derived. The top-level description of

the SoC, including the memories, interfaces and the black boxes of the processor and accelerators, is implemented using Verilog and VHDL. The accelerator cores and interface wrappers are implemented and fed to the wrapper generator. This generator is able to produce the various wrappers described in Section 4.6.1 based on the VHDL model of the accelerators.

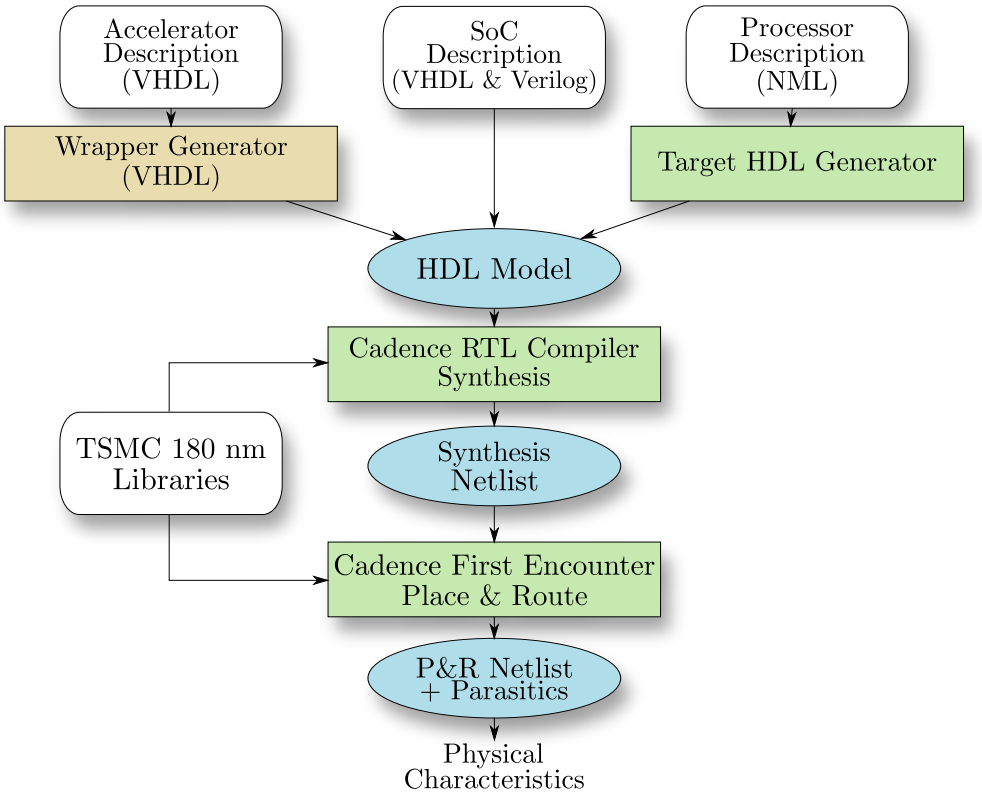


Figure 5.2: Synthesis tool flow.

The complete HDL model is then synthesized using the Cadence RTL Compiler 11.10 [7] using the TSMC 180 nm standard cell library. The synthesis result can be influenced using various user-defined constraints, like clock frequency, and rise and fall times. Furthermore, the RTL compiler is able to insert extra circuitry in order to enable clock gating, this option can be set for each module separately if necessary.

The gate-level netlist is used by the Cadence First Encounter 10.11 [7], which executes the place and route stage in various iterations. First, the floorplanning stage defines the physical dimensions of the chip as well as the placement for big blocks like memories and power supply lines. Second, the tool performs the placing of standard cells and inserts the clock tree. Therefore, several optimization iterations are necessary to meet the defined timing constrains. In the last stage, the tool connects the placed standard cells to the appropriate lines. Again, several iterations are executed in order to meet the timing requirements. Finally, a netlist of the placed and routed SoC including the extracted parasitics is saved. Figure 5.2 depicts the used synthesis flow.

5.2.2 Application Performance Flow

The application performance flow, as depicted in Figure 5.3, is used to obtain the performance of the application as well as to verify the functionality of the SoC.

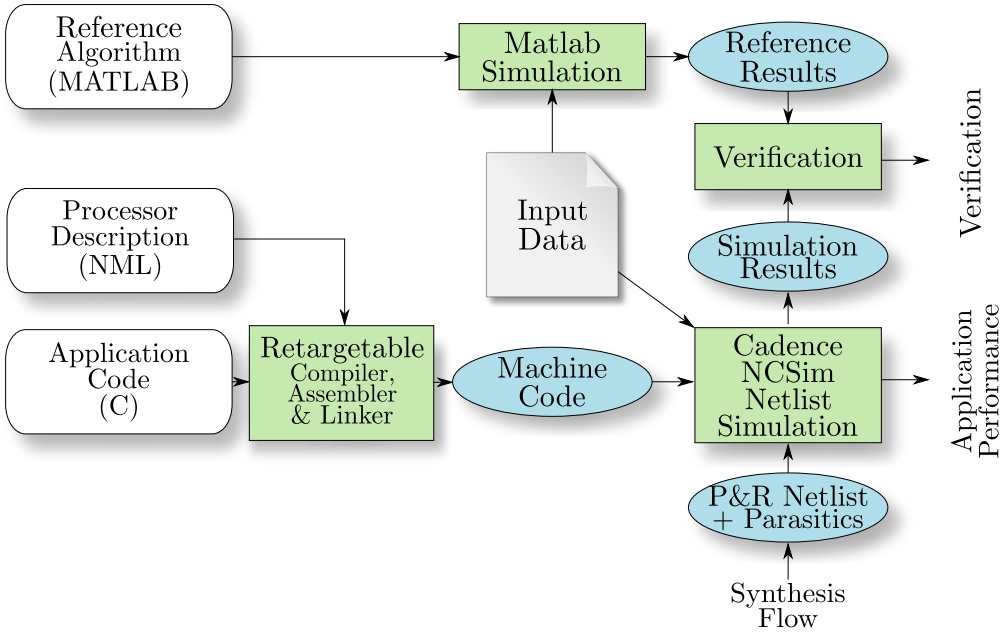


Figure 5.3: Application simulation tool flow.

The processor modeling language nML is used to describe the GPP model and provides the basis for the Target tool [74] to derive the retarget compiler, assembler and the linker. Furthermore, it would be possible to generate a cycle-accurate instruction-set simulator (ISS) based on this description. Due to the fact that the ASIP is equipped with external accelerators, however, the ISS is not able to simulate the whole application. Therefore, to simulate the system the netlist simulator Cadence NCSim [7] is used.

The compiled application is loaded into program memory before the simulation is executed. Input data from the AFE is sent to the digital back-end sample per sample within the testbench. NCSim uses the placed and routed netlist generated with the help of the synthesis flow described in the previous section.

To verify the simulation results, a reference model was implemented in MatLAB. A MatLAB simulation based on the same input data provides reference results so the functionality of the application executed on the ECGSoC can be confirmed.

NCSim is generally not designed for extracting the application performance parameters, and capturing cycle counts of single algorithms, especially, is much trickier than with an ISS. The only possibilities are to supervise the program counter or by adding a debugging variable into the data memory.

5.2.3 Power Estimation Flow

To determine the power consumption of the system, the power estimation flow is used. It combines the results of the synthesis flow and the application simulation flow.

A netlist simulation based on the placed and routed SoC is performed by executing the whole accurate R-peak search application. Therefore, the generated machine code as well as the input ECG signals are loaded into the testbench and processed by the ECGSoC.

Cadence NCSim is used to capture *value change data* (VCD), which contains the switching activity of all included cells. This information is used by the

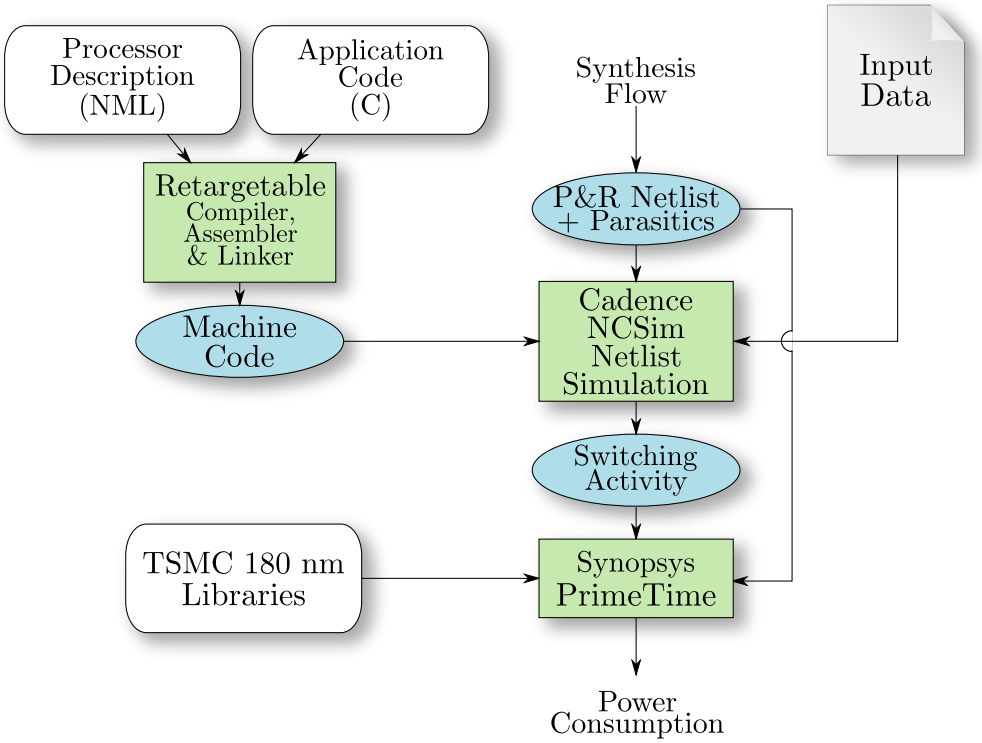


Figure 5.4: Power estimation tool flow.

power simulator which determines the consumed power. Synopsys PrimeTime 2011.12 [72] gathers the power values of the used standard cell from the TSMC 180 nm cells as well as the parasitics information from the placed and routed netlist.

To receive the most accurate power estimation it is necessary to use the placed and routed netlist for the simulation, otherwise the parasitics of the clock tree and the interconnect wires are estimated based on a general wire load model defined in Synopsys PrimeTime.

5.3 Accurate R-peak Detection Application

This section describes the R-peak detection application implemented at the dataflow-oriented ECGSoC in more detail. Furthermore, it provides insight at the algorithm-specific optimizations due to the proposed architecture.

5.3.1 MatLAB Reference Model

To simulate the functionality of the R-peak detection, a MatLAB system was created. This simulation model features all algorithms used within the application:

- Both **Motion Artifact Removal** algorithms, the PCA as well as the adaptive NLMS.
- The **Feature Highlighting** algorithm CWT, already implemented in a FIR-filter form.
- The **Beat Detection** algorithm using the variable threshold based on the QPD.

The model is used to produce reference results for the verification process. Furthermore, it allows a fast evaluation of the algorithm functionality. Therefore, the MatLAB model was used to explore the right set of coefficients for the adaptive NLMS algorithm as well as the attack and decay coefficients for the beat detection algorithm.

5.3.2 Architecture-Specific C

In order to execute the application on the ASIP efficiently, the algorithms were ported to the architecture-specific C dialect defined during processor modeling. The R-peak detection application developed for the original ECGSoC provided the basis of the actual application. The original application already used the processor in an efficient way.

The application was ported to the new processor applied in the proposed architecture. On the new system the application executes an increased amount of control instructions, due to the usage of external processing engines.

5.3.2.1 Compiler-Specific Optimizations

During application development it has been determined that the compiler produces a high amount of overhead when function calls are executed. Although the separation of algorithms into various sub-functions increases readability as well as reusability of the code, in order to reach maximum performance this has to be avoided using the given tool chain. Therefore, all computation-intense code snippets where implemented without using unnecessary function calls.

Furthermore, the compiler allows to specify a defined storage for variables, either a memory or a register. This allows to minimize load and store instructions by mapping frequently used variables to registers within computation-intense code snippets.

5.3.3 Motion Artifact Removal

5.3.3.1 Principal Component Analysis

As already mentioned in Section 4.5.1, the PCA is performed with the help of the ASIP. To allow effective execution of this algorithm, the original application provides an optimized implementation. Functions for matrix multiplication, eigenvector and eigenvalue computations sue the application-specific SIMD path of the processor to allow high performance and power efficiency.

5.3.3.2 Adaptive NLMS

Due to the applied adaptive NLMS accelerator, the application executed on the processor is just used for configuration purposes. The configuration, in the case of the new ECGSoC architecture, is to set the step size parameter. Furthermore, as introduced in Section 4.4.3, in the adaptive LMS use case scenario, the dedicated hardware performs the computation without the presence of the ASIP sample per sample.

5.3.4 Feature Extraction

The FIR accelerator provides an effective platform for executing the CWT algorithm. The application is responsible for the accelerator configuration, which, in the case of the proposed architecture, is the transfer of the filter coefficients to the dedicated hardware. Furthermore, in the PCA use case the application streams the motion artifact removal algorithms result to the dedicated hardware and receives the processed data.

One drawback of the CWT algorithm execution of the original system is that various coefficients and data samples are loaded multiple times. Therefore, the feature extraction stage produced a high amount of memory accesses, which lead to extensive power consumption. The proposed architecture, on the other hand, optimizes this behavior by using dedicated hardware, so each sample only has to be accessed once.

5.3.5 Beat Detection

The beat detection algorithm using an adaptive threshold, as described in Section 4.5.3, is executed by the ASIP. To allow effective computation, shift operations were used instead of multiplications when possible. Furthermore, threshold checking is optimized to use the least possible amount of cycles.

The reduce the number of memory accesses, the storage specify feature of the Target compiler is used. The number of function calls is also minimized to avoid unnecessary overhead produced during compilation.

5.4 Dataflow-Oriented ECGSoC Processing Architecture

The proposed architecture operates within the same conditions as the original system:

- **Operating Frequency:** 1 MHz
- **Supply Voltage:** 1.8 V
- **Process Technology:** TSMC 180 nm

In order to reach the lowest power solution it is not necessary to use the most advanced process technology [23]. Furthermore, a larger process technology is a valid choice for applications with such low performance requirements like the ECGSoC's, due to the increasing leakage power consumption in smaller processes.

5.4.1 SoC Top-Level Design

Applying the architectural modifications changes the system's top-level design significantly. All units which were implemented in the original system, described in Section 3.4, are still available and functional, with one exception. The direct memory access of the preprocessing unit is replaced with the proposed streaming based communication infrastructure. Figure 5.5 depicts the added units of the

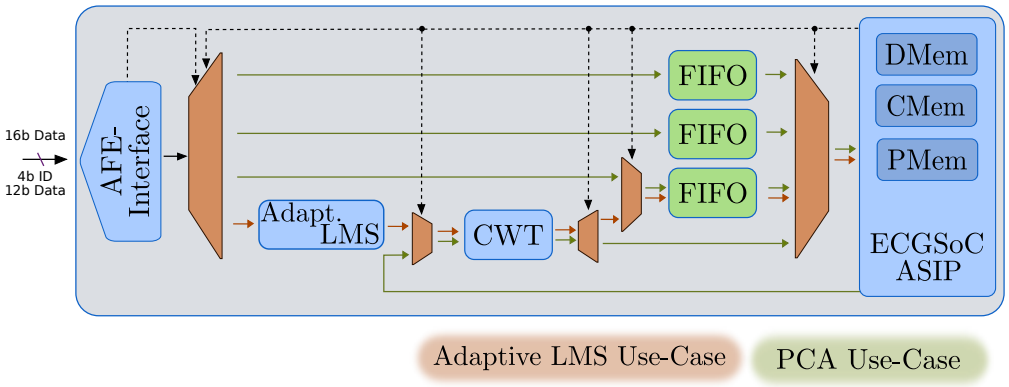


Figure 5.5: Top-level design of the proposed ECGSoC architecture.

proposed system, furthermore, it shows the communication paths necessary to execute the accurate R-peak detection application on the system. The datapath can be reconfigured for both application use cases. First, the adaptive LMS use case is depicted with red arrows, where the input data is processed by the LMS and the CWT accelerator and stored within the input buffer for further computation. Second, the PCA use case where all the ECG signals are stored in the FIFOs and after being processed by the ASIP are streamed to the CWT accelerator to execute the feature extraction algorithm. The configuration of

the communication channel can be influenced either by the deployed ASIP or the AFE-interface.

Additionally to the accelerators, it is necessary to implement three separate input buffers to store the gathered data of all ECG channels required by the PCA algorithm.

In addition to the mentioned changes, the applied clock gating concept is refined by adding suitable gates to the implemented memories.

5.4.2 Streaming-Based Communication

In Section 4.3.1 the necessary common denominator to allow reconfigurable streaming between two units within the proposed system is defined. The implemented connection uses 7 wires for the control signals and a parallel data bus. To allow a seamless integration to the ECGSoC's ASIP implementation, the data bus width is selected to be 128-bit. Therefore, the processor can process the data directly by using the SIMD datapath without overhead. Furthermore, collecting multiple samples and processing them at the same time reduces the necessary communication within the SoC.

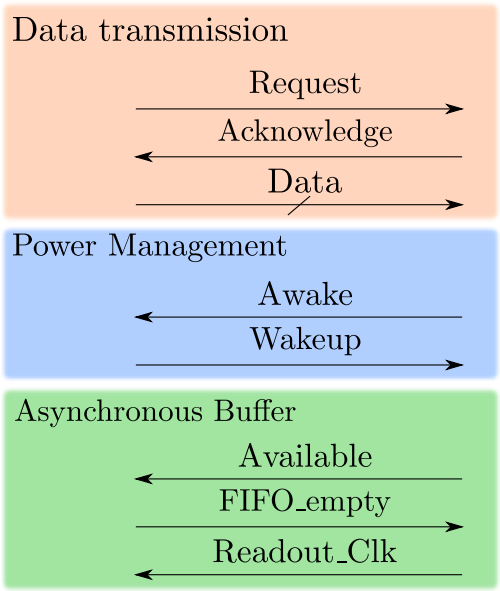


Figure 5.6: Signals necessary to implement streaming-based communication.

Figure 5.6 depicts the used data and controls signals subdivided by their purpose.

5.4.2.1 Data Transmission Signals

To transfer data between two components, a simple asynchronous data transmission scheme is used. Figure 5.7 shows such a request and acknowledge-based communication as implemented. The information is valid on the data bus from the moment the request is sent until the receiver acknowledges the reception.

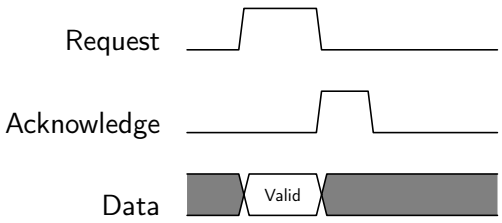


Figure 5.7: A single data transmission.

Such a communication requires that both communication components are awake and that the receiver is available for new data.

5.4.2.2 Power Management Signals

In order to ensure that all components are active, the wake-up and awake signals where implemented. This pair allows activation of the receiving component before a transmission is initiated. Therefore, the receiver's power management logic supervises the wake-up signal and reacts according to its state.

5.4.2.3 Asynchronous Buffer Signals

To prevent data loss and unnecessary delays, three control signals are used to control the small asynchronous buffers implemented within the communication interfaces. The functionality and usage of these signals will be explained in the following section.

5.4.3 Streaming Interface

The streaming interface is part of every implemented interface wrapper. This wrapper ensures the adoption of each module to the unified communication standard, as discussed in Section 4.6.1. The interface is deployed at the output

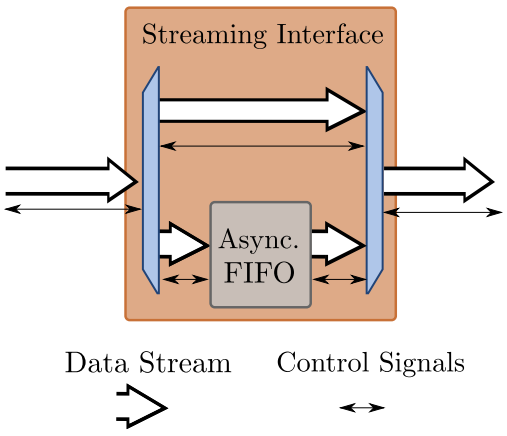


Figure 5.8: Implemented functionality within the streaming interface.

stream of each unit and handles the wake-up signal generation as well as the buffering of data in case of unavailable components or for modules using different operating frequencies.

To provide the necessary functionality the interface includes two data paths, as depicted in Figure 5.8. First, there is a direct connection between the communicating components which is used under normal operating conditions, more precisely, when both modules are awake and available for communication. Second, a data-path deploying an asynchronous FIFO to buffer the data if the receiving module is not awake or available is present. Furthermore, if the receiver operates at a lower frequency and is not able to handle all the sent data, this data path is able to buffer the information to prevent data loss.

Figure 5.9 depicts the case when data should be sent, but the receiver is not available. In a first step, the data is received by the interface, and due to the unavailable component the information is stored in the buffer and an acknowledge signal is sent back to source component. Therefore, the data source is available for further processing if necessary. The signal indicating the FIFO state falls, which indicates to the other component that data is present. Furthermore, after the FIFO is used, all communication will be handled by the data path deploying the buffer until the buffer is empty. Second, when the receiver becomes avail-

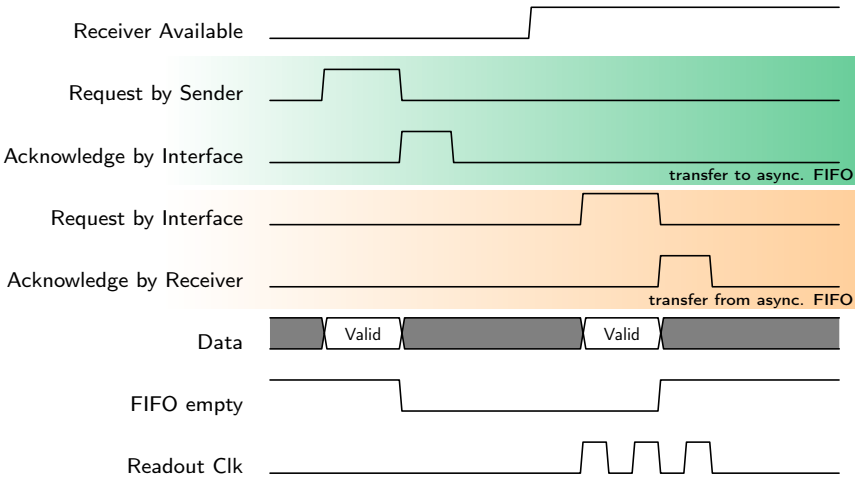


Figure 5.9: Data transfer across the asynchronous buffer.

able it reacts due to the low FIFO state signal and activates the readout clock signal. This clock signal initiates the interface to empty the buffer. Using the mentioned clock signal, the interface is able to send the data to the receiver even without the sender component. More precisely, the data source can even be inactive, as long it is awake. After emptying the buffer, the interface changes back to the directly connected data path.

In case of communication between components using different operating frequencies, it is required to adopt the pulse length from the acknowledge signals. More precisely, the pulse has to be long enough so that the counterpart will recognize the pulse. Therefore, the interface is equipped with some generic

parameters which allow adopting the pulse length.

5.4.3.1 ASIP Streaming Interface

In order to allow the ASIP to communicate via the implemented reconfigurable streaming-based communication architecture, it is necessary to equip the processor with additional interfaces to allow effective data transfers. Therefore two different interfaces are implemented:

The ASIP Stream Input Interface is based on 128-bit wide memory interface. This memory interface only allows read instructions. It is implemented with a 4-bit address vector, which is used to control a part of the reconfigurable datapath. Furthermore, the interface is able to rise an interrupt, so the processor is able to react to incoming data in every execution state. Unfortunately ,such interrupt routines require various additional instructions to save the actual processor state. Therefore, due to the predictability of the executed application, the interrupt is used to indicate the first transmission and afterwards the software itself checks if new data is available until the necessary amount of data is transferred. The necessary acknowledge information is gathered based on the read enable information of the ASIP.

The ASIP Stream Output Interface is also based on the generic memory interface provided by the Target [74] ASIP design suite. In contrast to the input interface, the output interface necessitates the ability to handle the acknowledge information. Therefore, the Target HDL generator has to be configured to generate the required functionality. This leads to additional wait states while the processor stands by until the acknowledge signal is received. The generated address vector is again used to reconfigure parts of the datapath.

5.4.4 Input Data buffer

A separate input buffer is deployed to store the gathered information for each necessary channel. The input buffer consists of an asynchronous FIFO and an interface wrapper. The wrapper adopts the buffer core to the generalized signals transferred within the streaming-based communication system. Figure 5.10 depicts the main building blocks as well as the implemented control signals of the FIFO. The buffer deploys a 128x128-bit *Standard Cell Memory* (SC-Memory). Two ports are used to provide information about the actual memory usage and two separate clock signals are used for read and write accesses. The FIFO uses internal registers to store actual read and write addresses to ensure prevention of data loss.

The used SC-Memory is a standard cell-based memory developed especially for low-voltage operations at IMEC-NL. This technology-independent memory offers competitive power numbers as well as high design flexibility. But in contrast to commercial memories the SC-Memory will use up to 10x more area.

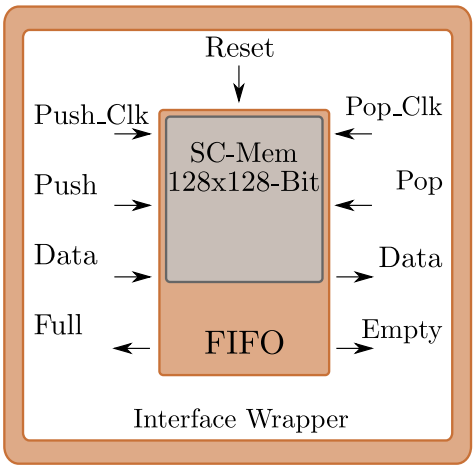


Figure 5.10: Input buffer overview.

5.4.5 Wrapper Generation

As mentioned in Section 4.6.1, parts of module architecture can be generated based on the core modules. More precisely, the *power-gated wrapper* and the *always-on wrapper* can be automatically generated. Therefore, a byproduct of this thesis is a small code generator used to generate the wrappers necessary for each module. Figure 5.11 gives an overview of the implemented code generation

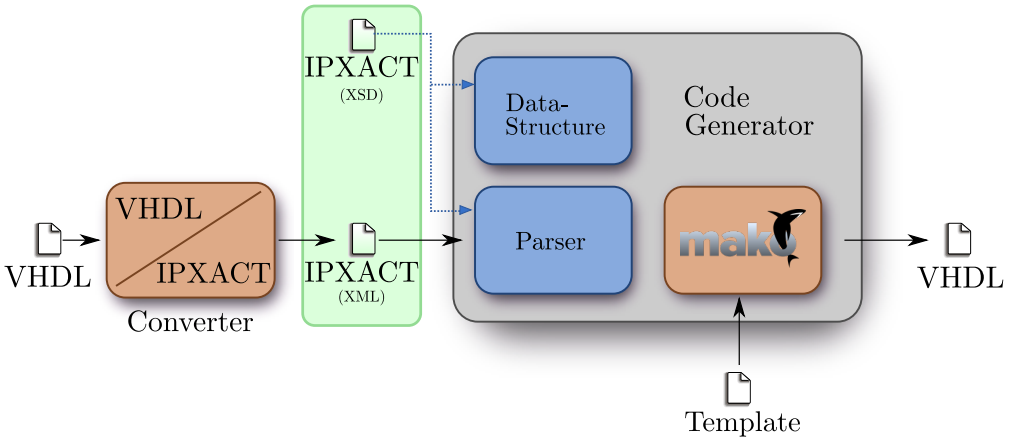


Figure 5.11: Code generator overview.

flow. The code generator is subdivided into tree parts, a parser to gather the information from the describing files, a data structure to store the information and a back-end, which generates the output based on template files. IP-XACT is used as input for the code generator, which is a vendor-neutral XML format created by the SPIRIT consortium [11] especially for describing and defining electronic components. Fortunately, this XML description includes a schema

description of the XML files, which builds the foundation of the parser and the data structure. Furthermore, using generateDS [46] allows automatically generating the necessary data structure including the parser for the code generator. To gather information from VHDL files and store it in the selected format a free VHDL to IPXACT converter [16] is used. As the basis for code generation, the back-end is provided by a template library called Mako [4]. The flexibility of the code generator is provided by the template files, within the templates, it is possible to define the required output which, therefore, offers a huge variety of possibilities.

In order to seamlessly integrate the wrapper generation within the used design-flow, various templates were required. Template files producing the required VHDL description as well as various configuration files to control the tool-flow have to be generated.

Using the wrapper generator offers the possibility to generate the required wrappers based on the HDL description of the modules.

5.4.6 Adaptive-LMS Accelerator

The accelerator designed to execute the adaptive NLMS algorithm is implemented as described in Section 4.6.2. To ensure correct functionality as well as the ability to power-gate the accelerator, the coefficient, error value and the input value tap delay line are implemented outside of the computational core.

In order to allow effective computation, the implemented accelerator is able to process one sample per clock cycle. Therefore, to resolve the data dependence within the algorithm, the coefficient update is executed on the falling edge of the clock cycle, while the output value is computed on the rising edge.

Due to the fact that the step size parameter (μ_0) is defined to be in a range between zero and two, it is necessary to convert the floating-point arithmetic used in the algorithm to an integer arithmetic which is compatible with the remaining hardware. Therefore, in order to achieve a high resolution, the step size parameter is shifted by 32-bits. This influences the required register size within the accelerator. Fortunately, the used synthesis tool removes unnecessary registers automatically and, therefore, ensures that the implemented accelerator uses the least area possible.

5.4.7 FIR Accelerator

Section 4.6.3 describes the design of a dedicated FIR accelerator implemented within the proposed ECGSoC architecture. This unit is used during the feature extraction stage of the R-peak application and executes the CWT algorithm.

The basis of the accelerator was provided by the DSP team of IMEC-NL. The generic FIR filter has been extended to fit into the module architecture defined for the proposed architecture. Therefore, the coefficients as well as the delay line are separated from the computational core to meet the defined module standard, as introduced in Section 4.6.1.

The implemented accelerator processes one sample per second and, therefore, allows effective CWT computation. The current CWT algorithm uses 63

coefficients and defines the size of the implemented FIR filter. Based on the defined input range of 16-bits, the resulting output size is 32-bits.

To furthermore reduce the power consumption, the provided FIR structure uses the *canonical signed digit* (CSD) number representation. Therefore, the coefficients have to be converted into the system by the application developer to suite the accelerator's implementation.

5.4.8 Implementation Procedure

5.4.8.1 HDL Model

The synthesizeable HDL model of the proposed architecture consist of various parts. The processor model is generated using the Target GO HDL generator as described in Section 5.2.1. This generator produces a Verilog model including all the user-defined instructions and the various specified memory interfaces. The original HDL model of the ECGSoC was also implemented using Verilog, the implemented changes were discussed in Section 5.4.1.

The additional added communication architecture and the accelerators are described using VHDL. All memories of the original system, as discussed in Section 3.4.2 are still used within the proposed architecture. Furthermore, a 2 kB SC-Memory is deployed in each of the three input buffers.

5.4.8.2 Synthesis and Place & Route

The gate level netlist is generated by using the Cadence RTL Compiler 11.10 and the TSMC 180 nm standard cell libraries. The user constrains applied during the synthesis are a defined operating frequency of one MHz and various SC-Memory-specific constrains.

To place and route the ECGSoCs architecture Cadence First Encounter 10.11 is used, this tool compromises floorplanning, placing of the cells and signal routing.

Floorplanning is the first step in order to place and route the design. Determining the optimal floorplan for the proposed architecture is an iterative process. Figure 5.12 depicts the final layout of the memories as well as the power ring and the power lanes added to supply the included macro cells. The four data memory cells are placed at the bottom of the floorplan, program and coefficient memory at the left boundary near the other macro cells, directly connected to the ASIP. The placement of the SC-Memory cells is more complicated due to the alignment of the connecting ports. Therefore, the floorplan must be chosen in such a way that enough area around the SC-Memories is standard cell free, so the routing stage has room to connect all ports of the cell. In order to avoid standard cell placement in the mentioned regions obstructs are placed around the memory locations.

Placement stage is used in order to place the standard cells within the layout, while avoiding the defined obstructs. After an initial placement run using the Encounters's default settings, various iterative optimization runs are executed

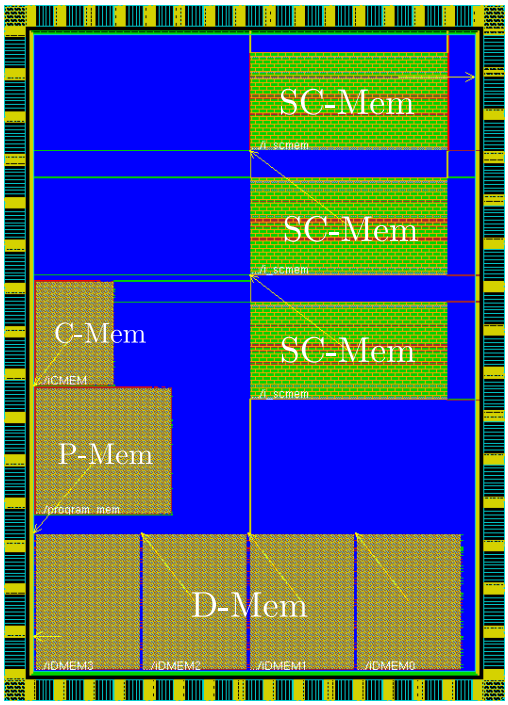


Figure 5.12: The proposed ECGSoC floorplan including memories and power lanes.

in order to minimize the setup- and hold time. After clock tree insertion, various iterations are necessary again to avoid timing violations and ensure a minimum of timing slacks.

Signal Routing stage is necessary in order to route all connections within the architecture. Therefore, a global detail route run is used, enabling the *timing drive* option of Cadence First Encounter. After routing, various incremental optimizations runs are executed in order to minimize the setup- and hold target slacks. Concluding the place and route stage, all empty space within the system is filled using filler capacities. The final chip layout is shown in Section 6.4.2.

Chapter 6

Experimental Results

6.1 Introduction

This chapter presents experimental results obtained by simulating the proposed ECGSoC architecture. Optimizations as well as implementation issues are evaluated and compared with the original system.

First, the evaluation of the baseline system, including the necessary methodology for extracting the energy used for communication, is presented. The power consumption of the original system executing the accurate R-peak detection application is presented in detail. Second, the simulation results of the proposed architecture are shown to verify the functionality of the system. Furthermore, synthesis and place and route results of the implemented architecture are provided. Concluding the chapter, the power consumption of this system is analyzed and discussed. Furthermore, the impact of the various applied optimizations is presented in detail.

6.2 ECGSoC V1.1 Evaluation

In order to compare the original system with the proposed architecture it is necessary to evaluate the baseline in detail. More precisely, the consumed power for various different tasks has to be extracted from the given power results. This thesis will focus on the required power for communication, as well as on the consumed energy by specific execution stages of the accurate R-peak detection application.

At this point it has to mentioned that the version 1.1 of the ECGSoC is already an improved version of the original system. In more detail, in this version, the clock gating concept of the SoC is already refined, i.e. clock gates were added to the memory cells of the baseline. This is necessary in order to allow a fair comparison between the different architectures. Otherwise, the results would be highly biased due to the varying clock gating concept and lead to wrong conclusions.

As mentioned in Section 3.3.3, within a DMA-based communication architecture, various different components contribute to the total consumed power caused by communication. While the power reports generated by the power

estimation tool flow (Section 5.2.3) easily allow to extract the consumed power of interfaces and controllers, a new methodology is required to derive the energy consumed by the bus itself as well as the memory consumed through communication. To capture this power consumption, a specialized set of tools is necessary which is introduced in the following section.

6.2.1 DMA-Bus Power Evaluation Methodology

Deriving the necessary power numbers from the available power reports and, therefore, allowing a detailed evaluation of the DMA-based communication system of the original system requires a new methodology. As a byproduct of this thesis, a new tool was implemented allowing further processing of the given power numbers by Synopsys PrimeTime. The tool is called "*Advanced PrimeTime Power Report Processing Suite*" (APPPS).

This tool enables one to derive the consumed power of the bus itself, furthermore, it allows a selective evaluation of the memory consumption to differentiate between energy used for communication as well as other tasks. Figure 6.1 de-

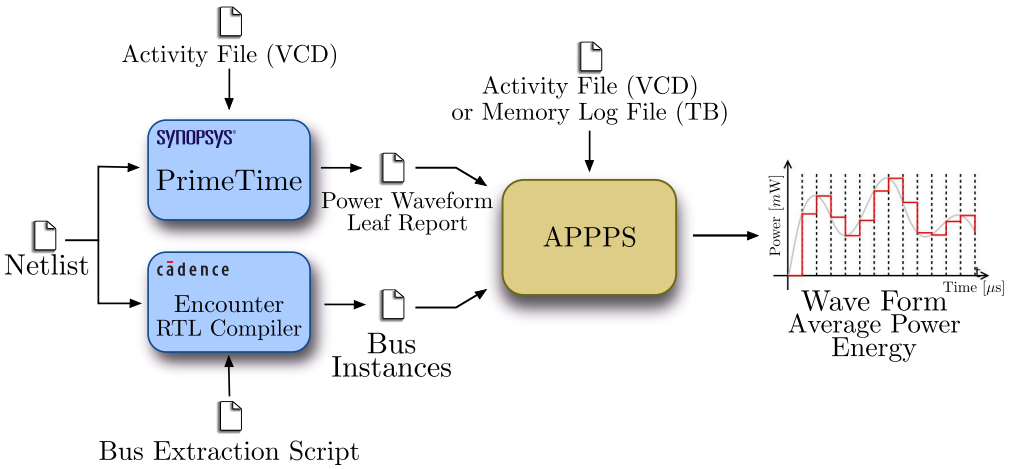


Figure 6.1: DMA bus consumption extraction tool flow.

picts the tool flow required to derive the consumed power of the bus, more precisely, the consumption of the wires and buffers. Based on the netlist the Cadence RTL Compiler is able to extract all instances connected to the DMA bus. Therefore, a tool-specific script was implemented executing an iterative search to identify the connected cells.

Simultaneously, the power estimation tool Synopsys PrimeTime calculates so-called leaf reports, which include a power waveform for each instance of the given netlist.

Based on the information gathered by these two tools, APPPS is able to compute a power wave form according to the consumed energy of the entire bus. Furthermore, it enables extraction of parameters like the average power or the total energy consumed by all instances required to allow communication via the DMA bus. In addition to the mentioned features, the implemented tool

is able to selectively extract consumed energy during specified intervals. These time spans can be defined either via a VCD file or a memory log file generated by the testbench. The log files furthermore contain information about the accessed addresses and, therefore, allow to extract consumed power based on defined address ranges, which is an important feature to subdivide the consumed power of the data memory

6.2.2 DMA Bus Memory Evaluation

In order to evaluate the a DMA-based communication system, it is necessary to take the power consumption caused by the shared memory into account. Therefore, the consumed power is divided into two general categories: power consumption is either caused by a read or write access to the memory cell or by the memory when it is in standby mode. Figure 6.2 depicts the subdivisions

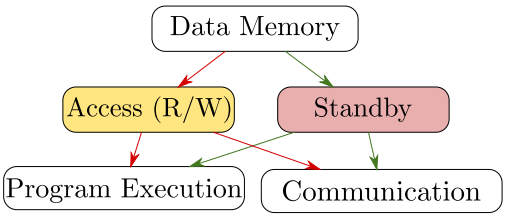


Figure 6.2: Subdivision of the memory power consumption.

of the memory power consumption. After dividing it into the two categories, the consumed power can be further assigned to its task. This evaluation differs between communication energy or energy used for program execution. To allow differentiation between these two tasks it is necessary to log the address of each memory access and further take the actual execution state of the application into account to ensure correct classification.

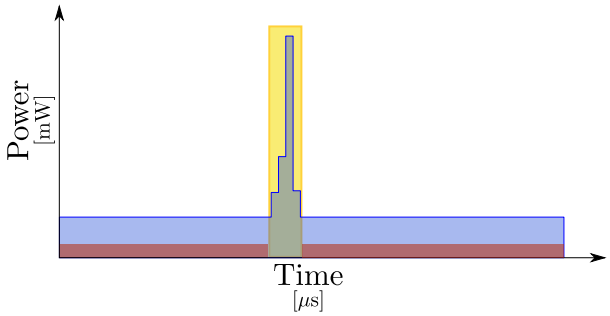


Figure 6.3: A single memory access.

Figure 6.3 depicts a detailed separation between the two main access and standby energy categories. The energy beneath the power peak caused by the memory access is part of the first category. On the other hand, the standby energy is divided according to the used memory size. So the fraction of the

standby energy assigned to a specific variable depends on the allocated memory size within the data memory.

6.2.3 Power Evaluation Results

6.2.3.1 Simulation Conditions

In order to allow a fair comparison between the different architectures, both systems were evaluated using the same boundary conditions. The conditions are subdivided into two categories.

Power Simulation Conditions: The used conditions for the power simulations are presented in Table 6.1. To get comparable results, these conditions apply for all results presented within this thesis. Although, as mentioned in Section 3.4.2, the original system is fully functional using a downscaled supply voltage of 1.2 V, it has been decided to use the nominal supply voltage of the process.

Table 6.1: Simulation conditions used for power evaluation.

Simulation Parameter	Value
Supply Voltage	1.8 V
Operating Frequency	1 MHz
Sampling Frequency	512 Hz
Simulation Parameter	Typical Case

Furthermore, all presented results refer to the top-level of the design, i.e. isolation cells as well as pads are neglected.

Application Conditions: In order to allow a fair comparison between the available use cases of the accurate R-peak detection application, the duty cycle of the PCA use case has been shortened. Table 6.2 gives an overview about the simulated duty cycle times within the original system.

Table 6.2: Application duty-cycle for all use cases (ECGSoC V1.1).

Application Use Case	Execution Time [ms]	Percent of Duty Cyle
Adaptive LMS	1023.97	100 %
Data Collection	994.12	97.08 %
Application Execution	29.84	2.92 %
PCA	1023.81	100 %
Data Collection	966.24	94.38 %
Application Execution	57.57	5.62 %

At this point has to mentioned, that a shorter duty cycle has no influence of the functionality of the application. Although it changes the lowest detectable heart rate, it does not influence the validity of the resulting power numbers.

All presented power results represent a full duty cycle of the already configured and processing system. So the results incorporate the repetitive task the system is executing.

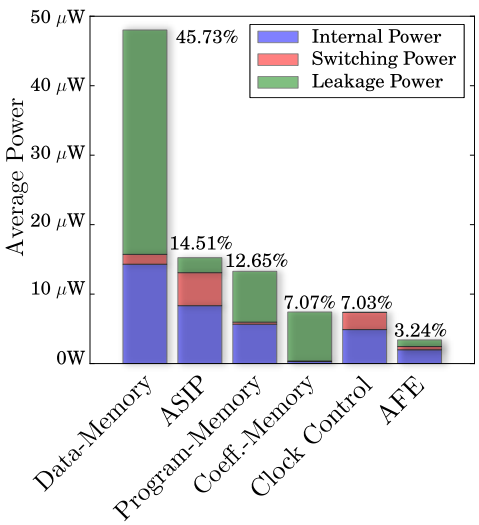
6.2.3.2 Power Consumption Overview

The average power consumption of the ECGSoC with the refined clock gating concept is shown for both use cases in Table 6.3. It can be seen, that, as expected, the PCA application consumes more power compared to the other use case. The required computational performance of the PCA algorithm is much higher compared to the adaptive LMS, therefore, these results are according to the expectations.

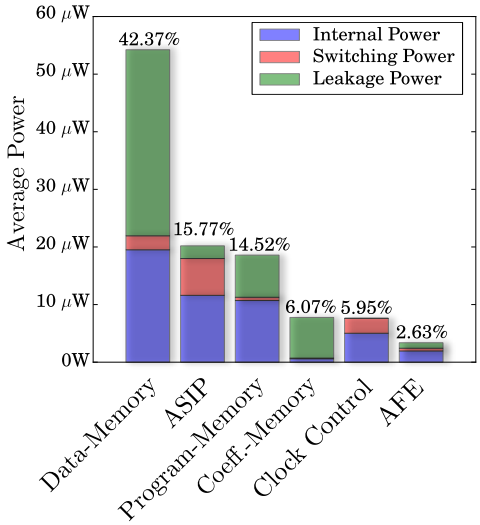
Table 6.3: Overall average power consumption.

Application Use Case	Avg. Power Consumption
Adaptive LMS	105.32 μW
PCA	128.04 μW

Figure 6.4 (a) and 6.4 (b) depict the top six consumers within the original ECGSoC design for each use case. In both cases, these six units cause approximately 90% of the entire power consumption. The plot identifies the distribution be-



6.4 (a): Adaptive LMS use case.



6.4 (b): PCA use case.

Figure 6.4: Top six consumers of the ECGSoC V1.1.

tween internal, switching and leakage power of each unit. This highlights that

an enormous amount of energy is consumed due to the leakage of the memory cells. Furthermore, the number located at the top of each bar represents the power fraction the unit consumes according to the total average power consumption. It can be seen, that the data memory is the main consumer within the architecture and, therefore, offers the biggest potential for improvements.

6.2.3.3 Power Consumption for Communication

The results in this section represent the energy consumed for communication, using the adaptive LMS use case. Figure 6.5 depicts the energy distribution within the original system. It can be seen, that the DMA bus, more precisely the wires and the buffer cells, only contribute to a negligible fraction of the total consumed energy within a duty-cycle. Furthermore, the plot shows that

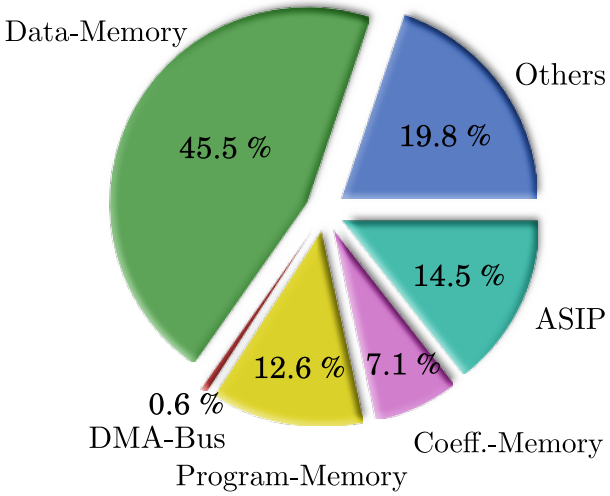


Figure 6.5: Energy distribution of the ECGSoC, including the DMA bus (Adaptive LMS Use-case: Total Energy 107.8 μJ).

the memory cells consume nearly three quarters of the entire energy. In order to evaluate the energy used for communication, Figure 6.6 depicts what causes the consumption within the data memory. As discussed in Section 6.2.2, the consumed energy within the data memory is subdivided into stand by and access energy. After this subdivision, the energy is assigned to a specific task: either program execution or communication. It can be seen, that 36.2 % of the energy consumed by the data memory is caused by communication.

Table 6.4 displays all components contributing to the energy used for communication within the ECGSoC. The sum of all components represent the energy necessary for communication, which, surprisingly, is in the order of magnitude of the ASIP energy consumption. The ASIP consumed 14.5% of the total energy, in contrast, the energy used for communication is 17.26% of the total energy consumption.

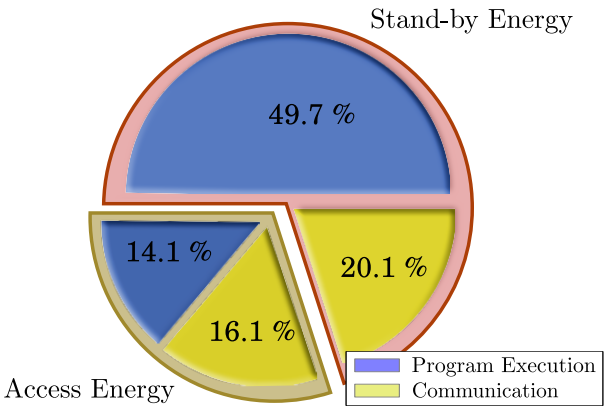


Figure 6.6: Energy distribution of the data memory (Adaptive LMS use case: total energy 49.11 μJ).

Table 6.4: Energy used for communication.

Components	Energy per duty cycle
Bus (Wires, Buffers)	0.61 μJ
Arbiter	0.05 μJ
Interfaces	0.19 μJ
Memory Cells	17.77 μJ
$\sum$	18.62 μJ

6.3 ECGSoC Algorithmic Verification

To verify the functionality and correctness of the executed R-peak detection application, the simulation results were verified by the means of the reference model implemented in MatLAB (see Section 5.3.1). While the MatLAB model is used to allow automatic verification, it is also possible to use it verify the functionality based on a graphical representation of the simulation results.

6.3.1 Adaptive LMS Verification

The adaptive LMS use case executes an adaptive NLMS algorithm in order to reduce the ECG motion artifacts. To highlight the occurring R-peaks the CWT algorithm is executed. In order to identify the peaks, the beat detection algorithm is used.

Figure 6.7 depicts the simulation results of the executed application. The plot represents the processing of four seconds of ECG data. The two plots at the top depict the input data, the ECG data including motion artifacts and the according measured electrode impedance. The resulting output of the motion artifact removal stage and the feature extraction stage are presented in the two plots at the bottom.

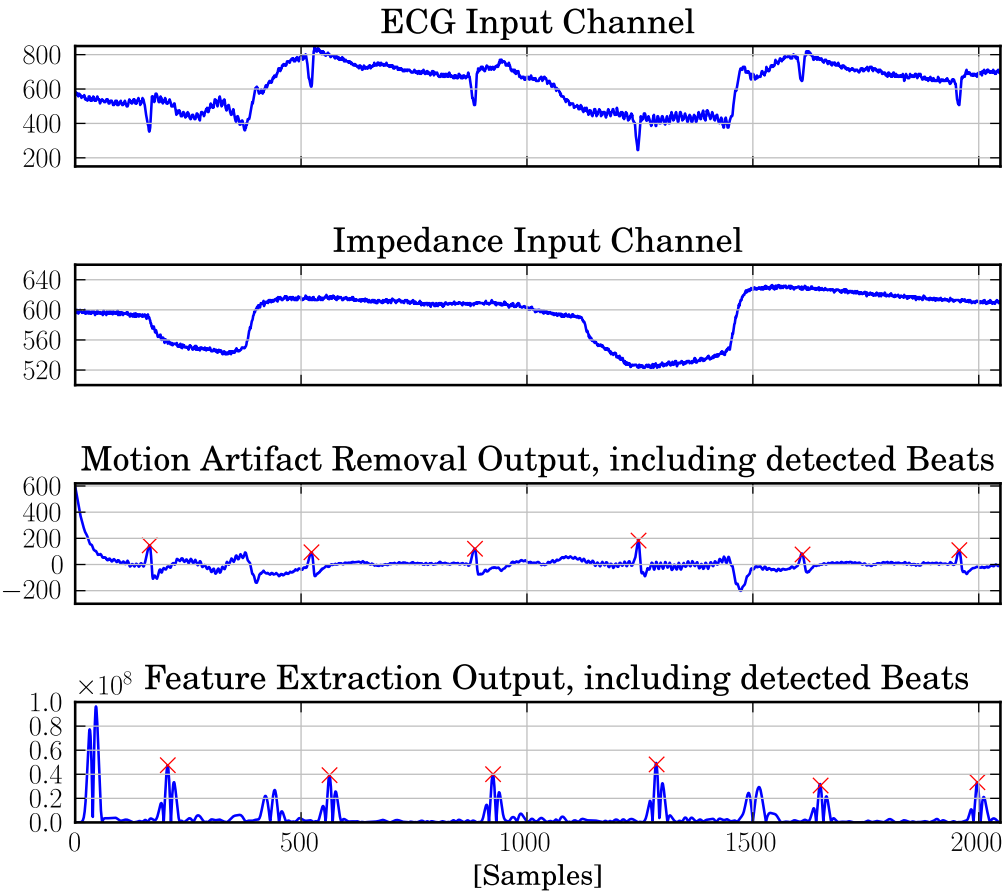


Figure 6.7: Simulation results executing the adaptive LMS use case.

These plots include the detected beat highlighted by a red x. The first peak shown in the CWT result is ignored by the beat detection algorithm. At application start, the adaptive motion artifact removal algorithm needs some time in order to settle and reach a transient state. Therefore, the beat detection algorithm ignores the first 100 samples.

Furthermore, a time offset can be observed between the detected beats in the different stages. The offset is caused by the CWT algorithm, the FIR structure adds a delay of half the filter length by definition. Fortunately, this does not affect the derived medical parameters like heart rate or applications like arrhythmia recognition.

6.3.2 PCA Verification

The PCA use case, in contrast to the adaptive LMS case executes the PCA algorithm in order to remove the motion artifacts.

Figure 6.8 depicts the simulation of the executed R-peak detection applications use case. Again, the 2048 processed samples represent four seconds of ECG data. The top three plots present the statistically independent ECG input

signals including motion artifacts. The resulting output of the various application execution stages is shown in the two bottom plots, including the detected beats. Again, the time offset between the stages can be observed.

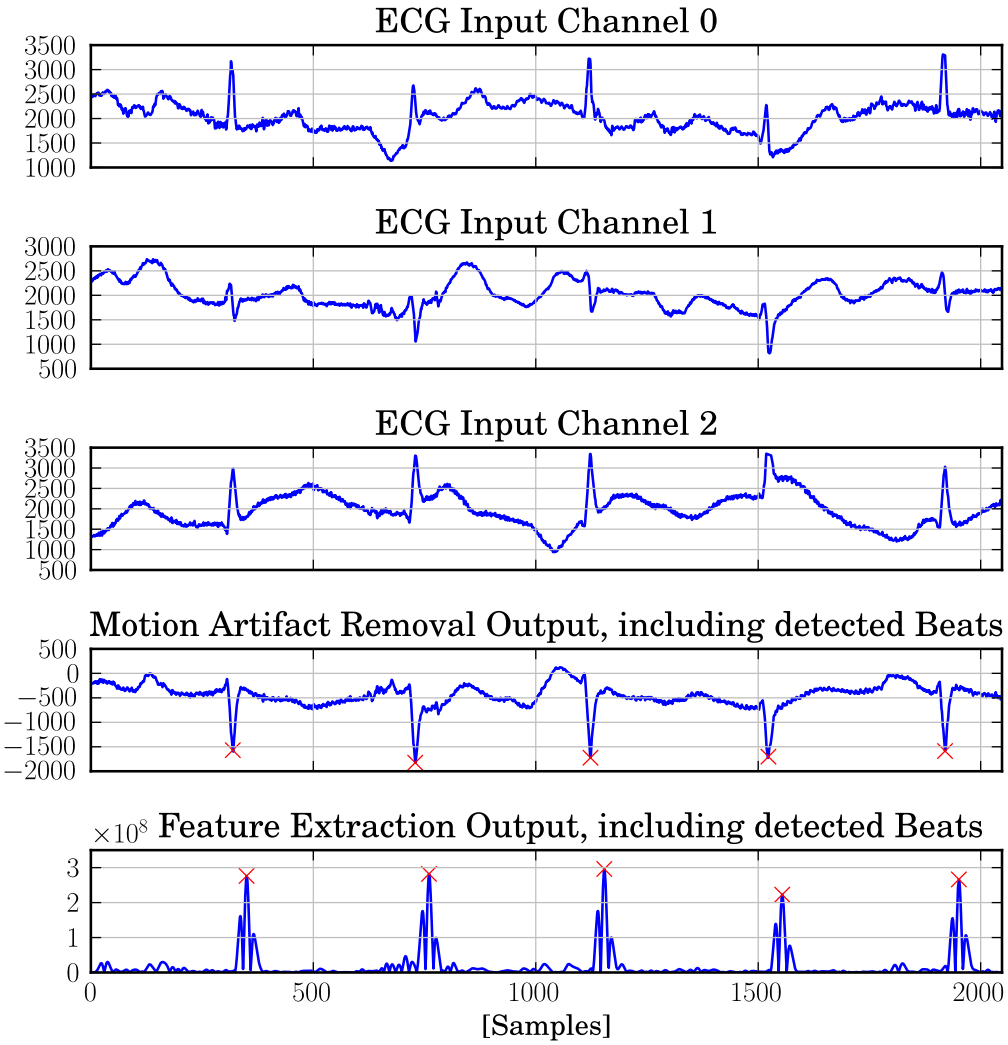


Figure 6.8: Simulation results executing the PCA use case.

The polarity of the motion artifact removal algorithms output depends on the eigenvalues of the input signals and can, therefore, vary. This does not influence the functionality of the feature extraction stage or the beat detection algorithm.

6.4 Physical Layout Results

6.4.1 Synthesis Results

The results in this section present the area required for the synthesized proposed ECGSoC architecture as outlined in Section 5.2.1. The synthesis of the whole architecture results in a required total cell area of 6745367 μm^2, which is an increase of 59.4 % compared to the original system. Table 6.5 gives an overview of the area distribution within the architecture. Almost 72 % of the cell area is required by the memories. The accelerators use approximately the same amount of area as the ASIP. The small increase of the ASIP area is caused by the additional memory interfaces. Due to the removal of the DMA path between data memory and AFE, the area requirements of the DMEM is slightly decreased.

Table 6.5: Area requirements of the proposed architecture as compared to the original system.

	Cell Area [μm^2]	NAND2 Eq.	%	Original System Cell Area [μm^2]	Difference [%]
Total	6745367	768195	**100.00**	4232535	+ 59.37
DMEM	1906025	217067	28.26	1907900	- 0.04
PMEM	652162	74271	9.67	652199	0.00
CMEM	276420	31480	4.10	276420	0.00
AISP	614658	70000	9.11	587363	+ 0.64
LMS Acc.	285545	32519	4.23	-	-
FIR Filter	302215	34418	4.48	-	-
FIFO	610696	69549	9.05	-	-
3x FIFOs	1832088	208647	27.16	-	-

The large increase of required area is a result of the SC-Memory usage within the input buffers. Although the FIFOs provide five times less storage space than the data memory, they occupy nearly the same amount of area.

6.4.1.1 Clock-Gating

Additionally to the top-level clock gates implemented by the designer, the synthesis tool was configured to add clock gates automatically as described in Section 2.6.3. This resulted in a total number of 303 clock gates, which are gating 91% of the deployed flip-flops.

6.4.2 Place and Route Results

The placement and routing of the standard cells was carried out as described in Section 5.4.8.2 and resulted in the layout presented in Figure 6.9. The utilization of the core area for the proposed architecture reached 75.94% using the specified floorplan and the place and route settings.

Table 6.6: Area requirements of the proposed architecture after place and route.

	Proposed Architecture $[mm^2]$	Original Architecture $[mm^2]$	Difference [%]
Core Area	8.81	6.3	39.84
Chip Area	10.92	10	9.20

Table 6.6 gives a comparison between the area requirements of the original system and the proposed architecture. Due to the fact that the chip size of the original system was defined by the number of pins, the required chip area only increased by roughly 10%.

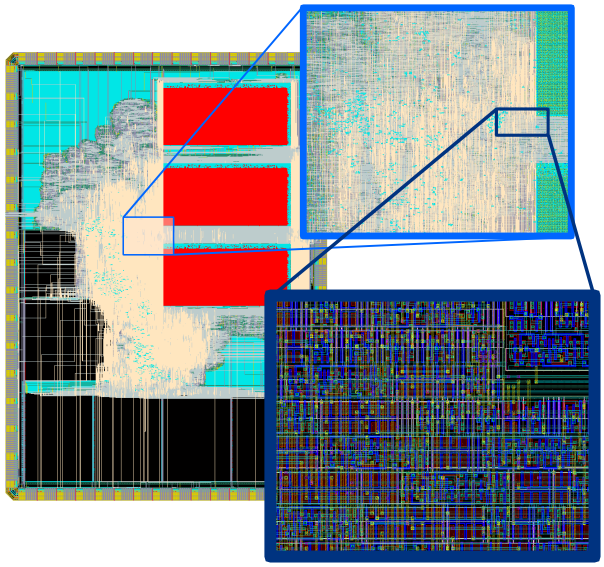


Figure 6.9: The dataflow-oriented ECGSoC after the place and route stage.

6.5 Power and Energy Consumption Results

The power consumption of the proposed architecture was determined using the power estimation tool flow described in Section 5.2.3. Furthermore, to ensure comparability, the power simulation conditions are the same as discussed in Section 6.2.3.1.

6.5.1 Power Consumption Overview

In order to provide an overview, Table 6.7 shows the energy consumed within one duty-cycle for both application use cases. Furthermore, the average power consumption is compared to the original system with the refined clock-gating

concept. The energy saving provided by the new architecture are in the range of 17-25%.

Table 6.7: Power consumption overview.

Use Case	Energy $[\mu J]$	Execution time [s]	Avg. Power $[\mu W]$	Original sys. Avg.Power $[\mu W]$	Difference [%]
Adapt. LMS	80.82	1.023	78.93	105.32	-25.06
PCA	107.85	1.023	105.37	128.04	-17.71

6.5.2 Detailed Power Consumption

Table 6.8 gives more details about power consumption within the proposed architecture. The seven top consumers are presented, including a comparison

Table 6.8: Power consumption details.

Component	Avg. Power $[\mu W]$	Original sys. Avg.Power $[\mu W]$	Difference [%]
Adaptive LMS Use Case			
Total	78.93	105.32	- 25.06
DMEM	33.04	47.96	- 14.17
PMEM	9.38	13.28	- 3.70
CMEM	7.18	7.43	- 0.24
ASIP	5.71	15.23	- 9.04
FIFOs	4.05	-	+ 3.85
FIR Filter	1.76	-	+ 1.67
NLMS acc.	1.29	-	+ 1.22
PCA Use Case			
Total	105.37	128.04	- 17.71
DMEM	41.08	54.22	- 10.26
PMEM	16.41	18.53	- 1.66
CMEM	7.62	7.77	- 0.12
ASIP	14.07	20.18	- 4.77
FIFOs	4.05	-	+ 3.16
FIR Filter	1.72	-	+ 1.34
NLMS acc.	1.48	-	+ 1.16

to the original system with the refined clock-gating concept. Although the data

memory is still the top consumer in the new architecture, the new design reduced the average power consumption of the data memory cells in a range of 10-14%.

The decreasing power consumption of the ASIP is due to the program execution time reduction based on accelerator usage. Furthermore, the additional power required by the accelerators to speed up the application is much less than the saved power. The reduction of the execution time also decreases the required power of the program and the coefficient memories.

6.5.3 Optimizations Overview

Within this thesis, various optimizations are applied and considered. This section should give an overview of the impact of these. The following section will explain the different optimizations in more detail.

Figure 6.10 depicts the results of various optimization stages for the ECG-SoC executing the adaptive LMS use case of the accurate R-peak detection application. Due to the fact that the memories applied in the system consume the main part of the whole power, the same optimization stages are depicted for low-power memory. The Synopsys Virage Memory [72] is a low-power memory

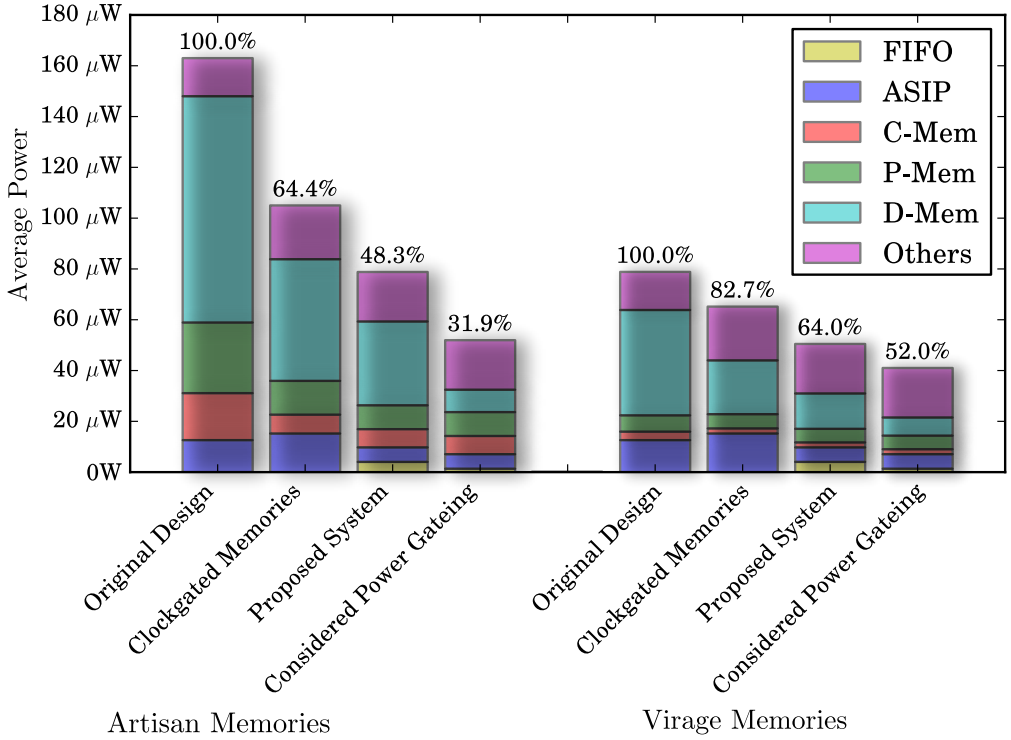


Figure 6.10: Average power consumption for the various optimization stages (adaptive LMS use case).

for the 180 nm TSMC process. In contrast to the applied Artisan memory, the Virage Memory cells have a lower maximal operating frequency. Therefore, the plot is separated into two groups, each optimization stage is calculated for the

different memories. At this point it has to be mentioned that the power consumption of the Virage memories is calculated based on a measured comparison between these two memories provided by IMEC-NL.

The first optimization step was the refined clock-gating concept of the original design, which has a high impact on the whole power consumption. Adding clock gates to the applied memories already saved 35% of the total power consumption. Using the proposed architecture further reduced the power consumption by another 25%. The concluding optimization step is to consider power gating. Using the last optimization step, the total power consumption can be reduced by a factor of three compared to the original design. Unfortunately, the TMSC 180 nm standard library does not include header and footer switches to implement power gating.

Using the Virage memories, the total power consumption could even be reduced by a factor of four.

Figure 6.11 depicts the impact of the various optimization stages for the ECGSoC executing the PCA use case of the accurate R-peak detection application. The impact of the architectural changes, as well as the impact of the

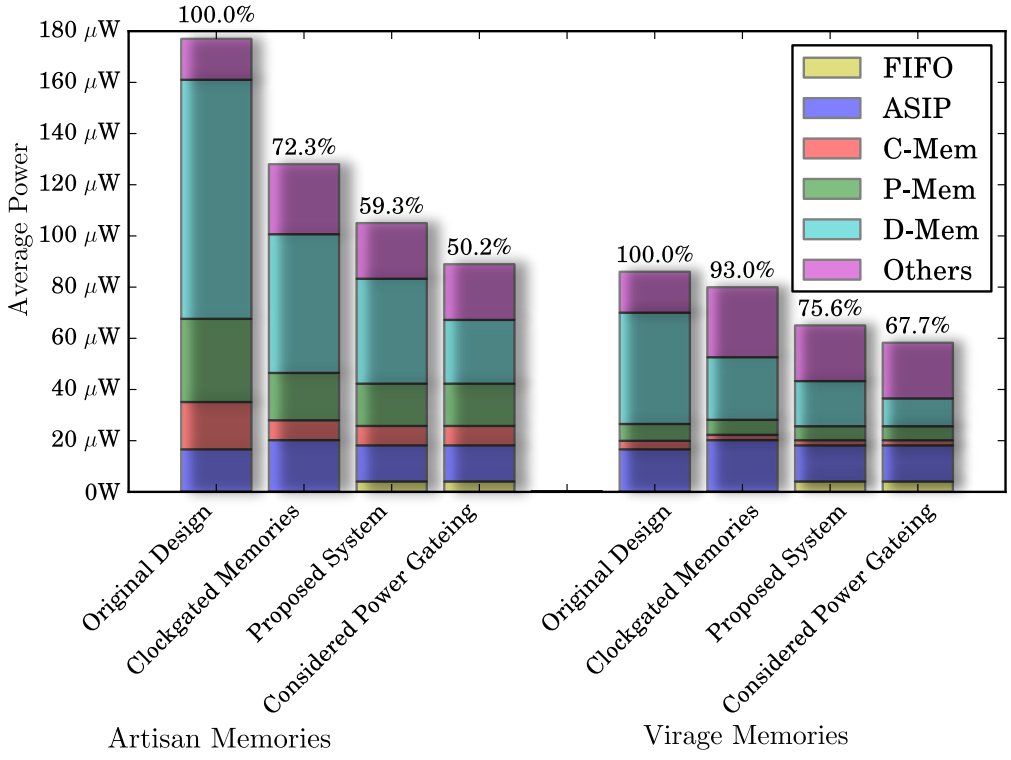


Figure 6.11: Average power consumption for the various optimization stages (PCA use case).

concluding power gating step is less compared to the use case discussed above. Due to the high memory usage of the PCA algorithm the effect of this optimization steps decreases. Although the impact is less, the total power consumption of the system is still reduced by a factor of two. Using the Virage memories

could even reduce the total power consumption by a factor of three.

6.5.4 Detailed Impact of Optimizations

The proposed architecture's impact on power and energy consumption was investigated for the usage of accelerators and applied communication architecture.

6.5.4.1 Impact of Accelerators

In order to provide a fair comparison between the different architectures, it is necessary to use another metric. Classical metrics like *energy per instruction* are not feasible to allow a direct comparison of an algorithm executed on an ASIP or an accelerator [22]. Therefore, the metric *energy per task* is used.

Adaptive NLMS accelerator The impact of the adaptive NLMS accelerator is depicted in Figure 6.12. Using the accelerator to execute the adaptive NLMS algorithm is 20 times more power-efficient than computing the algorithm with the deployed ASIP. The energy reduction using this accelerator represents 3.2%

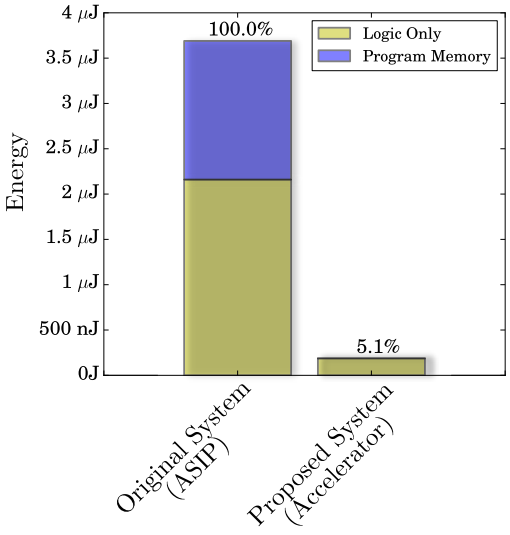


Figure 6.12: Impact of the adaptive NLMS accelerator.

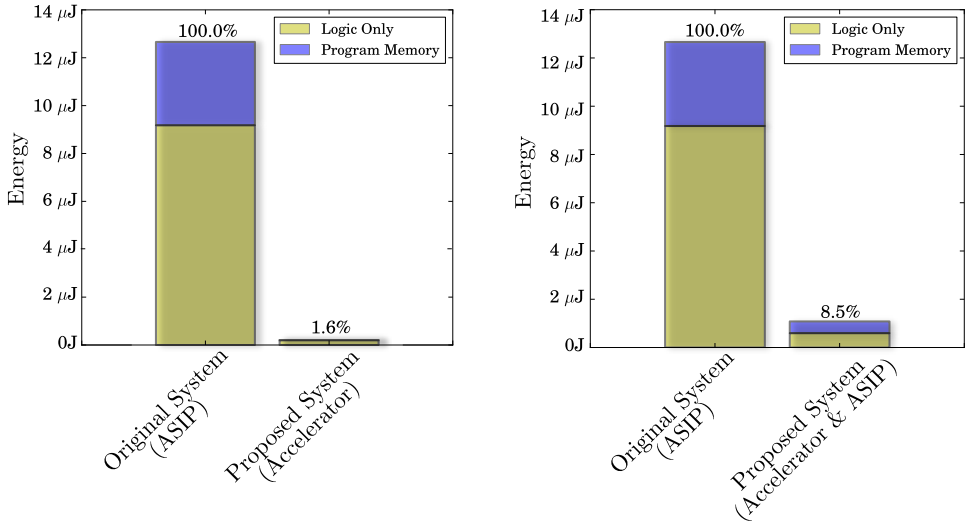
of the system's total power consumption when executing the adaptive LMS use case.

Table 6.9 compares the algorithms executed on the accelerators between the original system and the proposed architecture in terms of execution time and energy consumption.

FIR accelerator The impact of the implemented FIR filter is depicted in Figure 6.13. Both use cases require the execution of the feature extraction algorithm. The accelerator decreases the energy consumption for the adaptive LMS use case be a factor of 62.5 and, therefore, reduces the total consumption of

Table 6.9: Accelerators impact on energy consumption.

Algorithm	Proposed System		Original System		Difference	
	Execution Time [ms]	Energy [nJ]	Execution Time [ms]	Energy [μJ]	Time [%]	Energy [%]
Adapt. LMS	2.56	186.95	7.75	3.69	-66.97	-94.93
CWT	2.56	198.98	16.95	12.66	-84.90	-98.43



6.13 (a): Adaptive LMS use case.

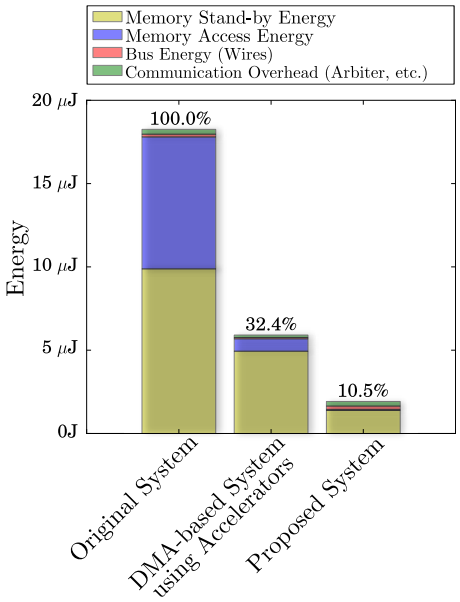
6.13 (b): PCA use-case.

Figure 6.13: Impact of the dedicated FIR filter.

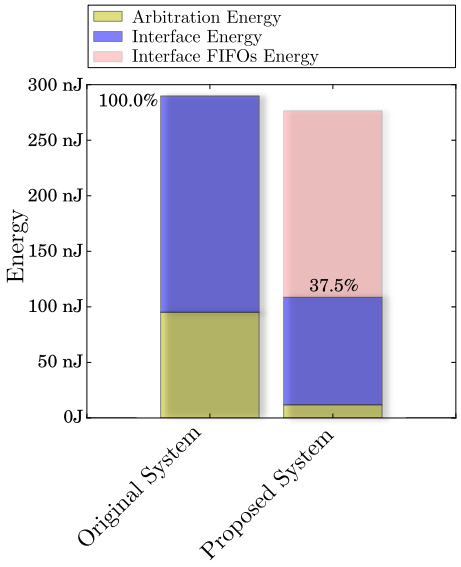
the application by 11.8%. Such a high reduction is possible due to the fact that the algorithm is executed without the presence of the ASIP. Within the PCA use case the ASIP is required to send and receive data to the accelerator. Therefore the, ASIP needs to be awake and execute the necessary instructions to fulfill the task. Figure 6.13 (b) depicts the impact of the dedicated FIR filter in the PCA use case. Under these circumstances, the accelerator reduces the energy per task by a factor of 11.7 and, therefore, reduces the total power consumption of the application by 8.8%.

6.5.4.2 Impact of the Communication Architecture

The impact of the dataflow-oriented ECGSoC on power and energy consumption was evaluated by the means of the adaptive LMS use case. Figure 6.14 (a) depicts the impact of the proposed communication architecture. The energy required for communication was reduced by a factor of 9.5, which represents a decrease of the total consumption of the application by 14.9%. Furthermore, using the accelerators within a DMA-based communication architecture, like the original system, would reduce the amount of necessary communication. Due to



6.14 (a): Total energy required for communication.



6.14 (b): Energy consumption caused by communicaiton overhead.

Figure 6.14: Impact of the proposed communication architecture.

the fact that the accelerators have built-in storage for the temporarily results, the number of memory accesses can be reduced. It has been estimated, that this would reduce the consumed energy roughly by a factor of three in contrast to the original system.

Figure 6.14 (b) shows the energy consumption caused by the communication overhead. Although the consumption through communication overhead is negligible compared to the total required energy, it shows that the energy required in the proposed architecture and the original system are approximately the same. But in contrast to the original system, the number of communicating hardware components has doubled in the proposed architecture. Furthermore, the interface FIFOs, as discussed in Section 5.4.3, are necessary to allow communication between components with different operating frequencies, which is not the case in the proposed ECGSoC implementation. Therefore, if the energy consumed by the FIFOs is neglected, the communication overhead's consumption would be reduced by a factor of 2.6.

6.6 Experimental Results Summery

This section summarizes the experimental results described within this thesis.

The core area of the proposed architecture requires approximately 40% more than the original system. Almost 72% of the used cell area is claimed by memory cells. Roughly 10% more chip area is required for the dataflow-oriented ECGSoC design, due to the fact that the original system's chip size was defined by the number of pins.

An extensive breakdown of the original system's power consumption has shown that the energy consumed for DMA-based communication was in the order of magnitude of the ASIP's energy consumption. The proposed architecture, while providing the same functionality and flexibility, reduced the average power consumption in a range of 17.7% to 25%, depending on the use case. The accelerators deployed in the proposed architecture were able to decrease the energy consumption of the adaptive LMS algorithm by a factor of 20, and of the CWT algorithm in a range of 12-62. The applied communication architecture reduced the energy consumption by a factor of 9.5, which represents a decrease of 14.9% of the total system's power consumption. Furthermore, considering power gating and low-power memories, the system's power consumption could be reduced up to a factor of 4, while executing the same application.

Chapter 7

Conclusions and Outlook

7.1 Conclusions

Within this thesis, a design concept exploration of a dataflow-oriented hardware-accelerated processing architecture applied in ultra-low-power biomedical sensor node applications was performed. The ambition of this exploration was to find a communication architecture that allows power-efficient data transfer and, furthermore, evaluate the impact of multiple accelerators deployed within a low-power WSN processing system. The concept was evaluated based on a provided ECG monitoring platform.

As the foundation of the design concept development, a detailed evaluation of the original system-on-chip was performed. An extensive breakdown of the system's energy consumption for the accurate R-peak detection application was carried out. The goal of this full-length consumption breakdown was the determination of the amount of energy used for data transfers, within the DMA-based communication architecture of the original system. Therefore, a new methodology was necessary in order to allow a detailed identification of the communication energy based on the available power estimation tools.

In order to allow a fair comparison between the original system and the proposed architecture, the clock gating concept of the provided platform was refined. Otherwise, the system comparison would have shown biased results, and lead to wrong conclusions.

Based on the information gathered from the evaluation, algorithms which where suitable for a power-efficient execution on hardware accelerators, were identified. Furthermore, accelerators able of executing the motion artifact removal as well as the feature extraction algorithms were designed. For high power efficiencies, these processing engines were designed to process one sample per cycle and, furthermore, allow application of state-of-the-art low-power techniques like clock and power gating.

A streaming-based communication concept to enable a dataflow-oriented architecture design was developed. Starting based on a reconfigurable unidirectional data stream, the idea of the proposed communication architecture was designed. A data path to stream the information and multiple control signals were used to allow information transfers between the components. Furthermore,

based on the control data of the communication channel, the execution states of the used components could be derived and, therefore, was used by the distributed power management unit to control the component-specific low-power features. In order to unify the communication architecture and include the power management control logic, a generalized module standard was defined. Using generatable, wrappers the various deployed modules connected via the proposed communication system were used to replace the former communication system of the ECG monitoring platform.

The final dataflow-oriented architecture was implemented, synthesized, as well as placed and routed, for a 180 nm process. Based on the physical layout, the system's power consumption was estimated. Using the resulting power figures, the impact of the different optimizations was explored.

The deployed accelerators were able to reduce the energy consumption of the adaptive LMS algorithm by a factor of 20, as well as the power consumption of the CWT algorithm within a range of 12-62, depending on the application use case. The proposed communication architecture has decreased the energy consumption by a factor of 9.5 compared to the original architecture with the refined clock gating concept. Although the same functionality and flexibility was provided, the average power consumption of the total system was reduced by 25% for the adaptive LMS use case and by 17.7% for the PCA use case. Considering power gating, the system would be able to run the same application while using a fourth of the energy as the provided ECG monitoring platform.

This thesis has shown that using hardware-accelerated processing architectures within a low-power processing system provides a balanced energy and flexibility trade-off. Computation extensive algorithms are executed on dedicated hardware, the deployed ASIP still executes software and, therefore, offers the necessary flexibility. Furthermore, it was shown that a reconfigurable streaming-based communication architecture is a valid choice for ultra-low-power processing systems.

7.2 Future Work

- **ASIP Streaming Interface:** An implementation of a specific streaming interface, which is further integrated into the ASIP as the current adapted memory interface, would lead to better source code readability. Furthermore the usage of custom instructions for interface handling can provide an implementation, where the usage of the comparatively slow interrupts is unnecessary.

- **Memories:** The evaluation has shown, that the data memory is the biggest consumer within the ECGSoC. Using an appropriate memory partition scheme can reduce the power dissipation of the system. Furthermore, due to the fact that the proposed architecture necessitates less memory space, the overall memory size can be reduced.

- **Generation of the Communication Network:** The reconfigurable datapath could be generated based on an abstract configuration file. An

extension of the implemented wrapper generator could allow an automatic generation based on the information of the modules and the configuration file. Furthermore, this could be integrated within the design tool-flow.

- **Off-the-shelf Configuration Bus:** To replace the custom configuration bus of the proposed system, a small lightweight off-the-shelf peripheral bus could be used. This would increase the reusability of the implemented modules as well as the readability of the application source code executed on the ASIP.
- **Smaller Process Technology:** Evaluating the proposed architecture concept within a smaller process technology could give better insight over the impact of the concept when the leakage consumption is dominating. Furthermore, an implementation using a smaller technology would allow a verification of the developed unified power-gateable module standard.

This page is intentionally left blank.

Appendix A

Abbreviations & Symbols

A.1 List of Abbreviations

ACK ..... Acknowledgment Signal
ADC ..... Analog-to-Digital Converter
AFE ..... Analog Front-End
AHB ..... Advanced High performance Bus
ALU ..... Arithmetic-Logic-Unit
AMBA ..... Advanced Microcontroller Bus Architecture
APB ..... Advanced Peripheral Bus
APPPS ..... Advanced PrimeTime Power Report Processing Suite
ASB ..... Advanced System Bus
ASIC ..... Application Specific Integrated Circuit
ASIP ..... Application Specific Instruction-set Processors
AXI ..... Advanced eXtensible Interface
BAN ..... Body Area Network
BD ..... Beat Detection
CPU ..... Central Processing Unit
CSD ..... Canonical Signed Digit
CWT ..... Continuous Wavelet Transform
DCR ..... Device Control Register
DMA ..... Direct Memory Access
DMAC ..... Direct Memory Access Controller
DSP ..... Digital Signal Processor
DTL ..... Device Transaction Level protocol
DVS ..... Dynamic Voltage Scaling
ECG ..... Electrocardiography
EEG ..... Electro-Encephalogram
EMG ..... Electromyography
FIFO ..... First-In, First-Out
FIR ..... Finite Impulse Response
GPP ..... General Purpose Processor
IP ..... Intellectual Property
ISS ..... Instruction-Set Simulator

ITRS ..... International Technology Roadmap for Semiconductors
LMS ..... Least Mean Square
MA ..... Motion Artifact
NLMS ..... Normalized LMS
NoC ..... Network-On-Chip
OCP ..... Open Core Protocol
OPB ..... On-chip Peripheral Bus
PC ..... Program Counter
PCA ..... Principal Component Analysis
PE ..... Processing Elements
PLB ..... Processor Local Bus
PM ..... Power Management
QPD ..... Quasi-Peak-Detection
REQ ..... Request Signal
SC-Memory ..... Standard Cell-Memory
SIMD ..... Single Instruction Multiple Data
SMP ..... Shared Memory Multiprocessor
UDVS ..... Ultra-Dynamic Voltage Scaling
ULP ..... Ultra-Low-Power
VCD ..... Value Change Data
VCI ..... Virtual Component Interface
WSN ..... Wireless Sensor Network

A.2 List of Symbols

$\mathbf{X}$ ..... Original data set
$\mathbf{Y}$ ..... Principal components
Ψ ..... Orthogonal linear transformation matrix

$x[k]$ ..... Input signal
$y[k]$ ..... Output signal
$d[k]$ ..... Desired signal
$e[k]$ ..... Error signal
$\vec{w}[k]$ ..... Coefficient vector
μ ..... Step size

$\psi^*(t)$ ..... Complex conjugated wavelet function
a, b ..... Dilation and location parameters

P ..... Dynamic ($P_{dynamic}$), static (P_{static}), total (P_{total}) power
C ..... Load (C_L), oxide (C_{ox}) capacitance
V_{dd} ..... Supply (V_dd), threshold (V_T), Gate-Source (V_{GS}), thermal (V_{th}) voltage
A ..... Activity factor
f_{Clk} ..... Clock frequency
I ..... Subthreshold leakage (I_{SUB}), gate leakage (I_{GATE}), gate induced drain leakage (I_{GIDL}), reversed bias junction leakage (I_{REV}) current

Appendix B

Additional Information

Table B.1 shows a comparison of all system architectures which where deployed in sensor nodes in the last decade. Table B.2 gives an overview of the performance rations of these systems.

B.1 Comparison of Ultra Low-Power Processors for Sensor Networks

Table B.1: Architecture comparison of ultra low-power processors for sensor networks.

System	Year	Architecture	Low-Power Technology	System Bus	Power Management
H. Zhang [88]	2000	ARM8 + 21 Accelerators 2x MAC, 2x ALU ,8x Mem. 8x Addressgen & 1x FPGA	-	two-level mash-structured reconfigurable	-
SmartDust [80]	2004	General Purpose	Separated clock	Multiple Busses (Custom)	-
E. Tell [76]	2005	DSP Optimised Ins. set Accelerators	-	core-configured Crossbar-switch	-
SNAP [15]	2005	Hardware Event Queue	Asynchronous	Custom Two Two level bus	Automatic
L. Nazhandali [57]	2005	General Purpose RISC	Sub-threshold	-	-
B. Zhai [86]	2006	General Purpose	Sub-threshold	-	-
Charm [68]	2006	General Purpose + DLL [1] Protocol Accelerator	Two Power Domains	-	centralised using PIF
S. Hanson [20]	2008	General Purpose Pipelined RISC	Sub-threshold	-	-
Phoenix [67]	2008	General Purpose	Near-Threshold	Asynchronous Protocol	Separate Unit
Sensium™ [83]	2008	8051 advanced architecture MAC protocol block	-	DMA Based Bus	-
J. Kwong [48]	2008	General Purpose	Sub-threshold	-	-

Continued on next page

System	Year	Architecture	Low-Power Technology	System Bus	Power Management
N. Ickes [31]	2008	GP + Accelerators for FIR and FFT	VDD-Gating Sub-threshold	DMA Based Bus	By Hand 12 Power Domains
S. C. Jocke [37]	2009	Custom PIC16C5X	Sub-threshold	-	-
J. Kwong [47]	2010	General Purpose + Accelerators for FIR CORDIC, FFT, MEDIAN	DVS, VDD-Gating Cock-Gating	DMA Based Bus	-
M. Hempstead [22]	2011	Accelerator / Event-Driven	VDD-Gating	Custom	By Hand
S. Hsu [27]	2011	AndesCore N903 RISC WFE + MLU + Cryto [2]	Power-Gating Dedicated Dutycycle Clock Subthreshold	AMBA AHB & DMA Based Bus	-
N. Ickes [32]	2011	ReISC [3]	Clock-Gating Sub-threshold	Crossbar	-
S. Sridhara [70]	2011	ARM Cortex M3 FFT Accelerator	Sub-threshold	DMA Based Bus	18 Power Domains
A. Y. Dogan [13]	2012	8x TamaRISC	Near-Threshold Clock-Gating Mem. VDD-Gating	2x Crossbars Mesh-of-Trees	-
N. Ickes [33]	2012	TI C64x+ Core 4-way VLIW	Standard Cell Design, VDD-Gating	DMA Based Bus	-
F. Zhang [87]	2012	DPM [4]+ Accelerators MCU, ECG, FIR	Event-Based, DVS, Clock & VDD Gating Sub-threshold Accelerators	Some parts are DMA-Based	Custom
BioDSP [6]	2010	Custom ASIP with 4 way SIMD	VDD-Gating	FIFO Point to Point	Automatic decentral

Continued on next page

System	Year	Architecture	Low-Power Technology	System Bus	Power Management
CoolBIO [29]	2011	NXP CoolFlux BSP Fixed Point DSP with Complex/SIMD data-path	Near-Threshold Voltage Domains VDD-Gating	Custom DMA Based Bus	By Hand 4 different Power Modes
ECG SoC [41]	2011	DSP 4 way 32-bit SIMD + Fir & AES-128 Accelerator	Separated Clocks Glock-gating	DMA Based Bus	-
TI MSP 430 [5]	2010	General Purpose RISC Architecture	Two Clock sources Clock-Gating	Memory data bus	5 Low power modes
NXP LPC 1110L [6]	2011	Cortex M0 Core	Clock-gating	AMBA AHB-Lite	3 Low Power modes

[1]DLL-Data Link Layer
[2]Wavelet-Extraction & Multi-Level-Compressor
[3]Reduced Energy Instruction Set Computer
[4]Digital Power Management Processor
[5]MSP430L092, MSP430C09x Mixed Signal Microcontroller 20 Sep 2010
[6]32-bit ARM Cortex-M0 microcontroller with 4KB flash, 1KB SRAM 2 Nov 2011

Table B.2: Performance ratio comparison of ultra low-power processors for sensor networks.

System	Year	Process	Datawidth	Memory	VDD [V]	Clk. [Mhz]	Average Energy
H. Zhang [88]	2000	250 nm	16-bit	24 kB	1	40	37.5-50 pJ/Cycle
SmartDust [80]	2004	250 nm	8-bit	3.125 kB	1	0.5	12 pJ/ins
E. Tell [76]	2005	180 nm	12-bit	17 kB	-	160	275 pJ/Cycle 785.5 pJ/Cycle
SNAP [15]	2005	180 nm	16-bit	8 kB	0.6	23	24 pJ/ins
L. Nazhandali [57]	2005	130 nm	-	-	0.235	0.182	1.38 pJ/ins
B. Zhai [86]	2006	130 nm	8-bit	0.25 kB	0.36	0.833	2.6 pJ/ins
Charm [68]	2006	130 nm	-	68 kB	1/0.3	8	96 pJ/ins 150 µW
S. Hanson [20]	2008	130 nm	8-bit	0.3125 kB	0.35	0.354	3.5 pJ/ins
Phoenix [67]	2008	180 nm	8-bit	0.41 kB	0.5	0.106	2.8 pJ/ins
Sensium™ [83]	2008	130 nm	-	64.75 kB	1	1	5 pJ/Cycle 500 pJ/Cycle
J. Kwong [48]	2008	65 nm	8-bit	128 kB	0.5	0.434	27.3 pJ/ins
N. Ickes [31]	2008	90 nm	16-bit	60 kB	0.45	4	10 pJ/cycle
S. C. Jocke [37]	2009	130 nm	8-bit	-	0.280	0.475	1.51 pJ/ins
J. Kwong [47]	2010	130 nm	16-bit	128 kB	0.5-1	0.1-10	25 pJ/Cycle
M. Hempstead [22]	2011	130 nm	8-bit	4 kB	0.55	12.5	0.44 pJ/ins
S. Hsu [27]	2011	90 nm	32-bit	8 kB	0.5	25	3.44 pJ/Cyc(RISC) 2.17 pJ/Cyc(Bio)
N. Ickes [32]	2011	65 nm	32-bit	16 kB	0.54 1.2	0.45 82.5	10.2 pJ/Cycle 41.7 pJ/Cycle [7]

Continued on next page

System	Year	Process	Datawidth	Memory	VDD [V]	Clk. [Mhz]	Average Energy
S. Sridhara [70]	2011	130 nm	32-bit	32 kB	0.5	0.007	29 pJ/Cycle
					1	5	114 pJ/Cycle
A. Y. Dogan [13]	2012	90 nm	16-bit	160 kB	1	-	15.6 pJ/Ops
N. Ickes [33]	2012	28 nm	32-bit	160 kB	1	587	192.5 pJ/Cycle
				Cache	0.34	3.6	200 pJ/Cycle
F. Zhang [87]	2012	130 mn	8-bit	5.5 kB	0.3-1.2	-	1.5 pJ/Op @ 0.5V
BioDSP [6]	2010	90 nm	32-bit	832 Kbit	0.7	100	10 pJ/Cycle
					1.2		177.2 pJ/Cycle
CoolBIO [29]	2011	90 nm	24-bit	2 Mbit	0.4	1	12.8 pJ/Cycle
					1.2	100	145 pJ/Cycle
ECG SoC [41]	2011	180 nm	32-bit	41 kB	1.2	1	$\approx$ 15 pJ/Cycle [8]
							$\approx$ 60 pJ/Cycle
TI MSP 430 [9]	2010	-	16-bit	4 kB	1.5/0.9	1	53 pJ/Cycle
							2.8 µW
NXP LPC 1110L [10]	2011	180 nm?	32-bit	5 kB	3.3	12	550 pJ/Cycle
							726 nW

[1] Cache disabled
[2] Only the digital part of the total consumption
[3] MSP430L092, MSP430C09x Mixed Signal Microcontroller 20 Sep 2010
[4] 32-bit ARM Cortex-M0 microcontroller with 4KB flash, 1KB SRAM 2 Nov 2011

Bibliography

[1] Giuseppe Anastasi et al. "An adaptive and low-latency power management protocol for wireless sensor networks". In: *Proceedings of the 4th ACM international workshop on Mobility management and wireless access*. MobiWac '06. Terromolinos, Spain: ACM, 2006, pp. 67–74.

[2] J.L. Ayala. *Communication Architectures for Systems-On-Chip*. Embedded Systems Series. CRC PressINC, 2011.

[3] C. Bachmann et al. "Low-power wireless sensor nodes for ubiquitous long-term biomedical signal monitoring". In: *Communications Magazine, IEEE* 50.1 (2012), pp. 20 –27.

[4] Michael Bayer. *Mako Templates for Python*. 2013. URL: `http://www.makotemplates.org/` (visited on 01/06/2013).

[5] Axel Böttcher. *Rechneraufbau und Rechnerarchitektur*. Springer-Verlag Berlin Heidelberg, 2006.

[6] B. Büsze et al. "Ultra Low Power programmable biomedical SoC for on-body ECG and EEG processing". In: *Solid State Circuits Conference (A-SSCC), 2010 IEEE Asian*. 2010, pp. 1 –4.

[7] Cadence. *Design Systems*. Cadence. 2012. URL: `http://www.cadence.com/` (visited on 12/05/2012).

[8] B.H. Calhoun et al. "Flexible Circuits and Architectures for Ultralow Power". In: *Proceedings of the IEEE* 98.2 (2010), pp. 267 –282.

[9] Francisco Castells et al. "Principal component analysis in ECG signal processing". In: *EURASIP J. Appl. Signal Process.* 2007.1 (Jan. 2007), pp. 98–98.

[10] Chee-Yee Chong and S.P. Kumar. "Sensor networks: evolution, opportunities, and challenges". In: *Proceedings of the IEEE* 91.8 (2003), pp. 1247 –1256.

[11] SPIRIT Consortium. *The Spirit Consortium Website*. 2013. URL: `http://www.spiritconsortium.org/` (visited on 01/06/2013).

[12] A. De Gloria and P. Faraboschi. "An evaluation system for application specific architectures". In: *Microprogramming and Microarchitecture. Micro 23. Proceedings of the 23rd Annual Workshop and Symposium., Workshop on*. 1990, pp. 80 –89.

[13] Ahmed Y. Dogan et al. "Multi-Core Architecture Design for Ultra-Low-Power Wearable Health Monitoring Systems". In: *Proc. of Design, Automation and Test in Europe (DATE '12), ACM and IEEE Press*. 2012.

[14] Virantha Ekanayake, Clinton Kelly IV, and Rajit Manohar. "An ultra low-power processor for sensor networks". In: *Proceedings of the 11th international conference on Architectural support for programming languages and operating systems*. ASPLOS-XI. Boston, MA, USA: ACM, 2004, pp. 27–36.

[15] Virantha N. Ekanayake, Clinton Kelly IV, and Rajit Manohar. "BitSNAP: Dynamic Significance Compression for a Low-Energy Sensor Network Asynchronous Processor". In: *Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 144–154.

[16] Kanai Lal Ghosh. *Miscellaneous EDA Utilities*. 2013. URL: `http://www.edautils.com` (visited on 01/06/2013).

[17] H. de Groot et al. "Human++: Key Challenges and Trade-offs in Embedded System Design for Personal Health Care". In: *Digital System Design (DSD), 2011 14th Euromicro Conference on*. 2011, pp. 4 –10.

[18] B. Gyselinckx et al. "Human++: Emerging Technology for Body Area Networks". In: *Very Large Scale Integration, 2006 IFIP International Conference on*. 2006, pp. 175 –180.

[19] Rehan Hameed et al. "Understanding sources of inefficiency in general-purpose chips". In: *SIGARCH Comput. Archit. News* 38.3 (June 2010), pp. 37–47.

[20] S. Hanson et al. "Exploring Variability and Performance in a Sub-200-mV Processor". In: *Solid-State Circuits, IEEE Journal of* 43.4 (2008), pp. 881 –891.

[21] S. Hanson et al. "Performance and Variability Optimization Strategies in a Sub-200mV, 3.5pJ/inst, 11nW Subthreshold Processor". In: *VLSI Circuits, 2007 IEEE Symposium on*. 2007, pp. 152 –153.

[22] M. Hempstead, D. Brooks, and G. Wei. "An Accelerator-Based Wireless Sensor Network Processor in 130 nm CMOS". In: *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on* 1.2 (2011), pp. 193 –202.

[23] M. Hempstead et al. "An ultra low power system architecture for sensor network applications". In: *Computer Architecture, 2005. ISCA '05. Proceedings. 32nd International Symposium on*. 2005, pp. 208 –219.

[24] Jason Lester Hill. "System Architecture for Wireless Sensor Networks". PhD thesis. 2003.

[25] R. Ho, K.W. Mai, and M.A. Horowitz. "The future of wires". In: *Proceedings of the IEEE* 89.4 (2001), pp. 490 –504.

[26] Cheng-Ta Hsieh and M. Pedram. "Architectural energy optimization by bus splitting". In: *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 21.4 (2002), pp. 408 –414.

[27] Shu-Yu Hsu et al. "A micropower biomedical signal processor for mobile healthcare applications". In: *Solid State Circuits Conference (A-SSCC), 2011 IEEE Asian*. 2011, pp. 301 –304.

[28] S. C. Huerta et al. "Review of DVS techniques to reduce power consumption of digital circuits". In: *Integrated Power Systems (CIPS), 2006 4th International Conference on* (2006), pp. 1 –6.

[29] J. Hulzink et al. "An Ultra Low Energy Biomedical Signal Processing System Operating at Near-Threshold". In: *Biomedical Circuits and Systems, IEEE Transactions on* 5.6 (2011), pp. 546 –554.

[30] Helmut Hutten. *Biomedizinische Technik: Diagnostik und bildgebende Verfahren*. Springer-Verlag, 1992.

[31] N. Ickes, D. Finchelstein, and A.P. Chandrakasan. "A 10-pJ/instruction, 4-MIPS micropower DSP for sensor applications". In: *Solid-State Circuits Conference, 2008. A-SSCC '08. IEEE Asian*. 2008, pp. 289 –292.

[32] N. Ickes et al. "A 10 pJ/cycle ultra-low-voltage 32-bit microprocessor system-on-chip". In: *ESSCIRC (ESSCIRC), 2011 Proceedings of the*. 2011, pp. 159 –162.

[33] N. Ickes et al. "A 28 nm 0.6 V Low Power DSP for Mobile Applications". In: *Solid-State Circuits, IEEE Journal of* 47.1 (2012), pp. 35 –46.

[34] Texas Instruments. 2013. URL: `http://www.ti.com/` (visited on 01/30/2013).

[35] ITRS. *International Technology Roadmap for Semiconductors 2011*. ITRS. 2012. URL: `http://www.itrs.net/` (visited on 11/12/2012).

[36] Ravi Iyer. "Accelerator-rich architectures: Implications, opportunities and challenges". In: *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*. 2012, pp. 106 –107.

[37] Steven C. Jocke et al. "A 2.6 muW sub-threshold mixed-signal ECG SoC". In: *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design*. ISLPED '09. San Fancisco, CA, USA: ACM, 2009, pp. 117–118.

[38] I.T. Jolliffe. *Principal Component Analysis (2nd ed)*. Springer, 2002.

[39] Stefanos Kaxiras and Margaret Martonosi. *Computer Architecture Techniques for Power-Efficiency*. 1st. Morgan and Claypool Publishers, 2008.

[40] M. Keating et al. *Low Power Methodology Manual: For System-on-Chip Design*. Springer-Verlag, 2008.

[41] Hyejung Kim et al. "A configurable and low-power mixed signal SoC for portable ECG monitoring applications". In: *VLSI Circuits (VLSIC), 2011 Symposium on*. 2011, pp. 142 –143.

[42] Hyejung Kim et al. "A low power ECG signal processor for ambulatory arrhythmia monitoring system". In: *VLSI Circuits (VLSIC), 2010 IEEE Symposium on*. 2010, pp. 19 –20.

[43] Hyejung Kim et al. *Design Document ECG-SoC*. Tech. rep. IMEC-NL, 2011.

[44] M. Kirst, B. Glauner, and J. Ottenbacher. "Using DWT for ECG motion artifact reduction with noise-correlating signals". In: *Engineering in Medicine and Biology Society,EMBC, 2011 Annual International Conference of the IEEE*. 2011, pp. 4804 –4807.

[45] Lakshman Krishnamurthy et al. "Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea". In: *Proceedings of the 3rd international conference on Embedded networked sensor systems*. SenSys '05. San Diego, California, USA: ACM, 2005, pp. 64–75.

[46] Dave Kuhlman. *generateDS – Generate Data Structures from XML Schema*. 2012. URL: `http://www.rexx.com/~dkuhlman/generateDS.html` (visited on 01/06/2013).

[47] J. Kwong and A.P. Chandrakasan. "An energy-efficient biomedical signal processing platform". In: *ESSCIRC, 2010 Proceedings of the*. 2010, pp. 526 –529.

[48] J. Kwong et al. "A 65nm Sub-Vt Microcontroller with Integrated SRAM and Switched-Capacitor DC-DC Converter". In: *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*. 2008, pp. 318 –616.

[49] K. Lahiri and A. Raghunathan. "Power analysis of system-level on-chip communication architectures". In: *Hardware/Software Codesign and System Synthesis, 2004. CODES + ISSS 2004. International Conference on*. 2004, pp. 236 –241.

[50] K. Lahiri, A. Raghunathan, and S. Dey. "Communication architecture based power management for battery efficient system design". In: *Design Automation Conference, 2002. Proceedings. 39th*. 2002, pp. 691 –696.

[51] I. Romero Legarreta et al. "Continuous Wavelet Transform Modulus Maxima Analysis of the Electrocardiogram: Beat Characterisation and Beat-to-Beat Measurement." In: *IJWMIP* 3.1 (2005), pp. 19–42.

[52] P. Lettieri and M.B. Srivastava. "A QoS-aware, energy-efficient wireless node architecture". In: *Mobile Multimedia Communications, 1999. (MoMuC '99) 1999 IEEE International Workshop on*. 1999, pp. 252 –261.

[53] M. Lyons et al. "The Accelerator Store framework for high-performance, low-power accelerator-based systems". In: *Computer Architecture Letters* 9.2 (2010), pp. 53 –56.

[54] Guoliang Ma and Hu He. "Design and implementation of an advanced DMA controller on AMBA-based SoC". In: *ASIC, 2009. ASICON '09. IEEE 8th International Conference on*. 2009, pp. 419 –422.

[55] G. Moschytz and M. Hofbauer. *Adaptive Filter*. Springer-Verlag, 2000.

[56] L. Nazhandali et al. "Energy optimization of subthreshold-voltage sensor network processors". In: *Computer Architecture, 2005. ISCA '05. Proceedings. 32nd International Symposium on*. 2005, pp. 197 –207.

[57] Leyla Nazhandali et al. "A second-generation sensor network processor with application-driven memory optimizations and out-of-order execution". In: *Proceedings of the 2005 international conference on Compilers, architectures and synthesis for embedded systems*. CASES '05. San Francisco, California, USA: ACM, 2005, pp. 249–256.

[58] Anja Niedermeier et al. "The challenges of implementing fine-grained power gating". In: *Proceedings of the 20th symposium on Great lakes symposium on VLSI*. GLSVLSI '10. Providence, Rhode Island, USA: ACM, 2010, pp. 361–364.

[59] G. Panic et al. "Sensor node processor for security applications". In: *Electronics, Circuits and Systems (ICECS), 2011 18th IEEE International Conference on*. 2011, pp. 81 –84.

[60] Sudeep Pasricha and Nikil Dutt. *On-Chip Communication Architectures: System on Chip Interconnect*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

[61] David A. Patterson and John L. Hennessy. *Computer Organization and Design, Fourth Edition, Fourth Edition: The Hardware/Software Interface (The Morgan Kaufmann Series in Computer Architecture and Design)*. 4th. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

[62] V. Raghunathan, M.B. Srivastava, and R.K. Gupta. "A survey of techniques for energy efficient on-chip communication". In: *Design Automation Conference, 2003. Proceedings*. 2003, pp. 900 –905.

[63] J. Robelly et al. "Implementation of recursive digital filters into vector SIMD DSP architectures". In: *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*. Vol. 5. 2004, V –165–8 vol.5.

[64] I. Romero. "PCA-based Noise Reduction in Ambulatory ECGs". In: *Computing in Cardiology*. Vol. Vol. 37. 2010, pp. 677–680.

[65] I. Romero et al. "Low-power robust beat detection in ambulatory cardiac monitoring". In: *Biomedical Circuits and Systems Conference, 2009. BioCAS 2009. IEEE*. 2009, pp. 249 –252.

[66] Iñaki Romero et al. "Motion artifact reduction in ambulatory ECG monitoring: an integrated system approach". In: *Proceedings of the 2nd Conference on Wireless Health*. WH '11. San Diego, California: ACM, 2011, 11:1–11:8.

[67] Mingoo Seok et al. "The Phoenix Processor: A 30pW platform for sensor applications". In: *VLSI Circuits, 2008 IEEE Symposium on*. 2008, pp. 188 –189.

[68] M. Sheets et al. "A Power-Managed Protocol Processor for Wireless Sensor Networks". In: *VLSI Circuits, 2006. Digest of Technical Papers. 2006 Symposium on*. 2006, pp. 212 –213.

[69] K. Slimani et al. "Low-Power Asynchronous Processors". In: *Low-Power Electronics Design*. Ed. by Christian Piguet. Vol. 1. CRC Press, Inc., 2004.

[70] S.R. Sridhara et al. "Microwatt Embedded Processor Platform for Medical System-on-Chip Applications". In: *Solid-State Circuits, IEEE Journal of* 46.4 (2011), pp. 721 –730.

[71] D. Sylvester and K. Keutzer. "A global wiring paradigm for deep submicron design". In: *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 19.2 (2000), pp. 242 –252.

[72] Synopsys. *Accelerating Innovation*. Synopsys. 2012. URL: `http://www.synopsys.com/` (visited on 12/05/2012).

[73] MingJian Tang and Jinli Cao. "An energy-efficient data-driven power management for wireless sensor networks". In: *Proceedings of the 5th workshop on Data management for sensor networks*. DMSN '08. Auckland, New Zealand: ACM, 2008, pp. 21–27.

[74] Target. *The ASIP Company*. Target. 2012. URL: `http://www.retarget.com/` (visited on 12/05/2012).

[75] Graz University of Technology Institute for Technical Informatics. *GeoWSN Project*. 2013. URL: `http://www.iti.tugraz.at` (visited on 01/10/2013).

[76] E. Tell, A. Nilsson, and D. Liu. "A low area and low power programmable baseband processor architecture". In: *System-on-Chip for Real-Time Applications, 2005. Proceedings. Fifth International Workshop on*. 2005, pp. 347 –351.

[77] D.A. Tong, K.A. Bartels, and K.S. Honeyager. "Adaptive reduction of motion artifact in the electrocardiogram". In: *Engineering in Medicine and Biology, 2002. 24th Annual Conference and the Annual Fall Meeting of the Biomedical Engineering Society EMBS/BMES Conference, 2002. Proceedings of the Second Joint*. Vol. 2. 2002, 1403 –1404 vol.2.

[78] Frank Vahid and Tony Givargis. *Embedded System Design: A Unified Hardware/Software Introduction*. 1st. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[79] Alice Wang, Benton H. Calhoun, and Anantha P. Chandrakasan. *Subthreshold Design for Ultra Low-Power Systems (Series on Integrated Circuits and Systems)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[80] B.A. Warneke and K.S.J. Pister. "An ultra-low energy microcontroller for Smart Dust wireless sensor networks". In: *Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International*. 2004, 316 –317 Vol.1.

[81] G. Werner-Allen et al. "Deploying a wireless sensor network on an active volcano". In: *Internet Computing, IEEE* 10.2 (2006), pp. 18 –25.

[82] Wikipedia. *Electrocardiography — Wikipedia, The Free Encyclopedia*. 2013. URL: `http://en.wikipedia.org/wiki/Electrocardiography` (visited on 01/30/2013).

[83] A.C.-W. Wong et al. "A 1V, Micropower System-on-Chip for Vital-Sign Monitoring in Wireless Body Sensor Networks". In: *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*. 2008, pp. 138 –602.

[84] R.F. Yazicioglu et al. "A 30 uW Analog Signal Processor ASIC for biomedical signal monitoring". In: *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*. 2010, pp. 124 –125.

[85] Rong Ye and Qiang Xu. "Learning-based power management for multi-core processors via idle period manipulation". In: *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*. 2012, pp. 115 – 120.

[86] Bo Zhai et al. "A 2.60pJ/Inst Subthreshold Sensor Processor for Optimal Energy Efficiency". In: *VLSI Circuits, 2006. Digest of Technical Papers. 2006 Symposium on*. 2006, pp. 154 –155.

[87] Fan Zhang et al. "A Batteryless 19uW MICS/ISM-Band Energy Harvesting Body Area Sensor Node SoC". In: *ISSCC*. San Francisco, 2012.

[88] H. Zhang et al. "A 1-V heterogeneous reconfigurable DSP IC for wireless baseband digital signal processing". In: *Solid-State Circuits, IEEE Journal of* 35.11 (2000), pp. 1697 –1704.