Christos Strydis

Universal Processor Architecture for Biomedical Implants

The SiMS Project

Universal Processor Architecture for Biomedical Implants

The SiMS Project

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.Ch.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen

op dinsdag 15 maart 2011 om 10:00 uur

door

Christos STRYDIS

Master in Computer Engineering
Technische Universiteit Delft
geboren te Athene, Griekenland

Dit proefschrift is goedgekeurd door de promotor:
Prof. dr. P.J. French

Copromotor:
Dr. G.N. Gaydajiev

Samenstelling promotiecommissie:

Rector Magnificus, voorzitter	Technische Universiteit Delft, NL
Prof. dr. P.J. French, promotor	Technische Universiteit Delft, NL
Dr. G.N. Gaydadjiev, copromotor	Technische Universiteit Delft, NL
Prof. dr. Y.N. Patt	University of Texas at Austin, USA
Prof. dr. A.V. Veidenbaum	University of California, Irvine, USA
Prof. dr. D.F.A.M.E. De Ridder	Universiteit Antwerpen, BE
Prof. dr. ir. H.J. Sips	Technische Universiteit Delft, NL
Dr. H.R. Lopuhaä	Technische Universiteit Delft, NL

CIP-DATA KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Christos Strydis

Universal Processor Architecture for Biomedical Implants — The SiMS Project
Delft: TU Delft, Faculty of Elektrotechniek, Wiskunde en Informatica - III
Thesis Technische Universiteit Delft. – With ref. –

Met samenvatting in het Nederlands.

ISBN 978-90-72298-14-0

Subject headings: implant, survey, taxonomy, benchmark, processor, cache, branch prediction, microarchitecture, instruction set, design-space exploration.

Printed in The Netherlands

Dedicated
to Stamatis who made sure this thesis would be started;
to Georgi who made sure this thesis would be concluded.

Universal Processor Architecture for Biomedical Implants

The SiMS Project

Christos Strydis

Abstract

HEALTHCARE in the 21st century is changing rapidly. In advanced countries, in particular, healthcare is moving from a public to a more personalized nature. However, the costs of healthcare worldwide are increasing every year. Better use of technology can and should be used to get control of these costs. At the same time, implants have clearly benefitted from the astounding technology-miniaturization trends of late, boasting smaller sizes, lower power consumption and increased performance of the transistor devices. However, such advances do not come for free. Adverse effects in current implant designs are being witnessed, such as increasing power consumption, absence of design for reliability and highly application-specific nature. Operating under the assumption that implants will constitute an important means towards improved, personal healthcare and, in view of the aforementioned design phenomena, we believe that a new paradigm in implant design is required. This dissertation establishes the concept of Smart implantable Medical Systems (SiMS). SiMS is a systematic approach – a framework – for providing biomedical researchers and, hopefully, industry with a toolbox of ready-to-use, highly reliable implant sub-systems and models in order to construct optimal implants for various medical applications. The SiMS framework has to guarantee essential attributes, such as high dependability, modular design, ultra-low power consumption and miniature size. Having defined the SiMS framework, this dissertation is, then, concerned with exploring the optimal microarchitectural details of a crucial SiMS component: the SiMS processor. Contrary to the current state of the art, this processor aspires to be a new universal, low-power and low-cost processor and capable of efficiently serving a wide range of diverse implant applications.

Acknowledgements

There is a very good reason why acknowledgements are always found in the beginning of a book. They tell the private story of how the book you are holding in your hands came to be. A number of people have contributed in a number of ways to its contents and looks. Had they not helped when they did, I am certain this work would have never reached its present state of completion.

So, I would like to start off by thanking my co-promotor Georgi Gaydadjiev for the sustained insight and support he provided throughout the hardships of this thesis work. His being the yin, I would also like to thank my yang – my late promotor Stamatis Vassiliadis for his initial encouragement and *ab initio* backing of my research. God rest his soul. I will always consider both men as my mentors and my friends. I am also deeply grateful to my thesis-committee member Rik Lopuhaä for offering unstinting support on statistics, and on a very short notice at that. This thesis would have been far from complete without his contribution.

As far as the tool flow that was necessary for this thesis is concerned, the work would have come to an early standstill, had it not been for the valuable contributions of Stefanos Kaxiras, Margaret Martonosi and Gilberto Contreras who have been instrumental in providing me with an excellent power-simulation tool (XTREM) and all support I could ask. I am also indebted to Demid Borodin for his assistance in setting up and understanding the chaos that is called cross-compilation, as well as to Pepijn de Langen, Filipa Duarte and Vlad-Mihai Sima for their helpful insights on the matter. I would also like to thank Kyriakos Stavrou for his prompt, catalytic help with the CACTI cache-simulation tool and Carlo Galuzzi specifically for his formulating of the XTREM power analysis and, overall, for his valuable comments and help throughout. Special thanks are due to Bert Meijs, Erik de Vries and Eef Hartman – the past and present CE-Laboratory system administrators – for their inexplicable sympathy for my outrageous technical requests and for their excellent support in setting up and operating the custom computer cluster needed for our DSE experiments. Without them, this work would not have taken shape.

Furthermore, I remain truly grateful to all those people who responded to my persistent requests or unequivocal threats by offering precious resources, insights or even strong criticism. To name a few: Christopher Sadler for providing the sources of his intriguing SLZW compression algorithm, Peter Cross for providing the original sources for the excellent DMU application and limitless support, Johnny Ray Sears for providing resources on his biotelemetric heart-valve monitor, Pietro Valdastri for providing the assembly sources of running in his implantable telemetry platform system and Niki Frantzeskaki for her valuable help on statistics and otherwise. Also, Christoforos Kachris, Dhara Dave and Zhu Di for the fruitful collaboration we have achieved. Furthermore, I greatly appreciate and wish to thank Dimitris Theodoropoulos for his helpful insights on data encryption and computer architecture, Sebastian Isaza, Yiannis Sourdis, Daniele Ludovici and Lotfi Mhamdi for being the boxsack for me to hone my work on; and Nicolae & Nicole Stefu for giving me a nudge in the right direction when I needed it the most. My thanks are also due to both the CE-group secretaries Lidwina Tromp and Monique Tromp for their exceptional administrative assistance throughout my PhD years. My deep thanks are also due to Ana Laura Santos for – among other things – putting together the elegant design that is the book cover of this dissertation document. What is more, I wish to acknowledge the authorities responsible for (partially) funding this thesis work: the ICT Delft Research Centre (DRC-ICT) of the Delft University of Technology and Google Inc.. It was their timely aid that ensured the completion of this thesis.

Last but certainly not least, I feel the need to deeply thank my wife Olga and my parents Stefanos and Aristea for their immeasurable emotional as well as downright material support, for their patience and faith throughout the course of this research effort. I am – and always will be – in their debt.

Christos Delft, The Netherlands, March 2011

Table of Contents

List of Tables

List of Figures

List of Acronyms

ADC Analog-to-Digital Converter

ASIC Application-Specific Integrated Circuit

BAN Body-area Network

BiCMOS Bipolar CMOS

BioMEMS Special MEMS class wherein biological matter is manipulated

BP Blood Pressure

BPRED Branch PREDiction

CMOS Complementary Metal-Oxide Semiconductor

CNS Central Nervous System

COTS Commercial-Of-The-Shelf

CRC Cyclic-Redundancy Check

DAC Digital-to-Analog Converter

DBS Deep-Brain Stimulation

DDM Denervated, Degenerated Muscle

DFR Design For Reliability

DSE Design-Space Exploration

ECG Electro-Cardio-Graphy

EDA [in Computer Eng.] Electronic Design Automation

EDA [in Statistics] Exploratory Data Analysis

EMC Electro-Magnetic Compliance

EMG Electro-Myo-Graphy

EMI Electro-Magnetic Interference

ENG Electro-Neuro-Graphy

EOG Electro-Oculo-Graphy

EPIC Explicitly-Parallel-Instruction Computing

FES Functional Electrical Stimulation

FNS Functional, Neuromuscular Stimulation

FP Floating Point

FSM Finite-State Machine

GA Genetic Algorithm

HDL Hardware Description Language

HLL High-Level Language

HPC Hardware Performance Counter

IC Integrated Circuit

ICD Intra-Cardiac Defibrillator

ICP Intra-Cranial Pressure

ILP Instruction-Level Parallelism

IP Intellectual Property

IPC Instructions per Cycle

ISA Instruction-Set Architecture

LSK Load-Shift Keying

MEMS Micro-Electro-Mechanical System

NPA Non-Programmable Accelerator

NPM New Public Management

PCB Printed-Circuit Board

PCC Processing and/or Controlling Core

PD Parkinson's Disease

PDA Personal Digital Assistant

PNS Peripheral Nervous System

QoS Quality of Service

QRS The most visually obvious deflections seen on a typical ECG

RAS Return-Address Stack

RF Radio Frequency

RSNA Renal-Sympathetic-Nerve Activity

SC Switched-Capacitor

SCS Spinal-Cord Stimulation

SE Standard Error (σ)

SNR Signal-to-Noise Ratio

SoC System-on-Chip

ULP Ultra-Low Power

UWB Ultra-Wide-Band

VLIW Very-Long-Instruction Word

WSN Wireless Sensor Network

1 Introduction

COME to think of it, human inventions have always aspired to imitate nature. And shameless imitation it has been to the accuracy allowed by each era's conquered scientific knowledge and available technical means. This mimicry has at times been conscious and at others not. The Da Vinci Glider [ca. 1500] (Figure 1.1) is an infamous instance of such an attempt to imitate bird flight. Da Vinci went all the way from the theory of flight to – actually – designing flying machines and attempting to fly them. The fact that the original design proved to be too heavy for flight is besides the point.

1.1 Background

It may sound far-fetched at first but (bio)medical implants are just another instance of mimicking nature throughout human history. Prosthetic body parts such as wooden limbs and reeds for looking and listening inside the human body are proved to have been used by ancient Egyptians as early as 3,000 years B.C.. In recent years, scientific knowledge underpinned by astounding technological achievements – the "technical means" previously mentioned – in fields such as microelectronics technology as well as material science and more have led to the fully implantable pacemaker. The implant was developed in 1958 and 1959 (of course, not microelectronic at the time) by Wilson Greatbatch and William M. Chardack and has been the first device to be implanted successfully into the human body and to operate seamlessly for long periods of time – modern pacemakers feature an in-body lifetime of a decade or longer. Perhaps more importantly, it has also acted as a catalyst on the general public closed-mindedness against biomedical implants. Ever since the pacemaker, a plethora of other biomedical implants has also been proposed for solving various medical problems, however, only a few of them have made it to the market

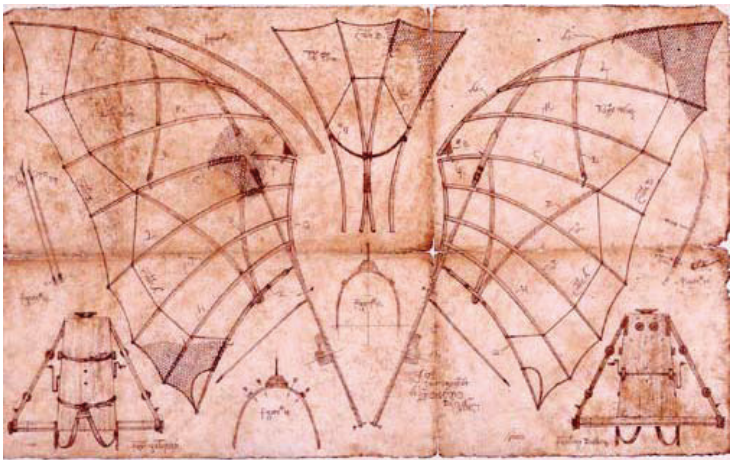


Figure 1.1: *The Da Vinci glider design.*

such as the widespread cochlear implants and the deep-brain stimulators for treating Parkinson's Disease.

1.2 Problem overview

1.2.1 Socioeconomic trends

Signified by the advent of the first, implantable pacemaker in the late fifties, biomedical microelectronic implants have evolved rapidly over the last 50 years, as much in application scope as in sophistication. However, a close look at the current implant market reveals relatively few successfully commercialized ideas. The phenomenon can be attributed to a number of reasons:

- the – until recent times – limited knowledge of many aspects of the human body combined with the large inter-patient genetic variability when coming to human pathoses, which makes it difficult and risky to define clear-cut implantable solutions to address all of them; and
- the practical limitations posed so far by technology; for instance, transistor sizes have not been small enough to allow for complex implant designs to fit inside the tight constraints of the human body. Besides, substituting sophisticated, physiological structures with artificial ones is a highly complex and largely interdisciplinary task.

While problems such as the above have been allowing for slow yet steady implant progress, current *socioeconomic trends* manifesting in modern societies may not permit us even this pace any longer. To illustrate, in advanced countries the following cascading trends are currently being observed:

- Population is aging through a net reduction in birth rates combined with an increase in life expectancy;
- Healthcare costs are growing out of proportion; and
- Higher demands for betterment of quality of life are placed (health, fitness, convenience etc.).

Healthcare in the 21st century is changing rapidly. In advanced countries, in particular, healthcare is moving from a public to a more personalized nature [3, 26]. The costs of healthcare worldwide are increasing every year. In *the Netherlands*, in particular, the government is trying to keep the health insurances affordable for all citizens by periodically reorganizing the system. Since healthcare spending always increases at a much faster rate than the average income, such practices work only for a limited period of time. The rising healthcare costs, in combination with population aging (i.e. more potential customers for the healthcare system), form a tough challenge for modern societies.

Such socioeconomic trends have given birth to the notion of *personalized healthcare*. The term introduces a new approach to effective healthcare – as far as economics go, at least – whereby default hospitalization and generic treatment of patients will be discouraged and supplanted by *patient-targeted* prognosis, diagnosis and, mainly, treatment. It is highly conceivable that *technology* will be the vehicle for enabling personalized healthcare. Similar trends have already been witnessed in the cell-phone and portable-computing technological revolutions.

Better use of technology – and, in our case, implants – can and should be used to get control of healthcare costs. For example, continuous monitoring of physiological parameters can be used instead of occasional meetings with the doctor. Having an up-to-date and complete picture of the changes in a patients condition will enable disease prognosis, which by definition is more effective and less costly than disease treatment. It should be stressed that such technology will be used not only for high-risk or chronic patients, but also for general lower-risk patients over periods of normal activity in their home or work environment.

Implants have clearly benefitted from the astounding recent *technology-miniaturization* trends [66], boasting smaller sizes, lower power consumption and increased performance of the transistor devices. Simply put, while the human-body dimensions have not changed, microelectronics dimensions have – by proportion – shrank to such an extent that modern implants are becoming sufficiently sophisticated and small so as to treat various human pathoses, even at the most constrained parts of the human body. It is this practical property along with their wider acceptance in modern societies that is making them a primary technological driver towards personalized healthcare.

1.2.2 Implant-device trends

Although biomedical implants may be (advertised to be) a primary vehicle for the brave, new era in personalized healthcare, it has come to our attention – and we have verified it through an extensive survey – implant design is undergoing a shift itself. Namely, implants are gradually moving from application-specific, rigid (e.g. FSM-based) systems to more flexible (e.g. μProcessor-based) ones. This implies that, in the near future, implant functionality will be based on executed software (written in some high-level, established language like C) rather than on hardwired circuits. However, this turn of events does not come for free; adverse effects are being witnessed:

- Implant *power budgets are increasing* over time, even though transistor dimensions are shrinking and implemented device functionality is not overtly complex;
- Implants exhibit serious *absence of design for reliability*. Software-based, ad-hoc reliability techniques have been replacing inherently reliable implant designs over the years. For a field of highly-mission critical embedded systems where human lives and high costs are involved, this poses a significant problem;
- Product development is still *highly application-specific*, even though implant designs are becoming more structured. Already established product cases such as the family of pacemakers introduced by Medtronic, Inc. [91], where previous design expertise is (re)used to enhance the next device version, are currently the exception.

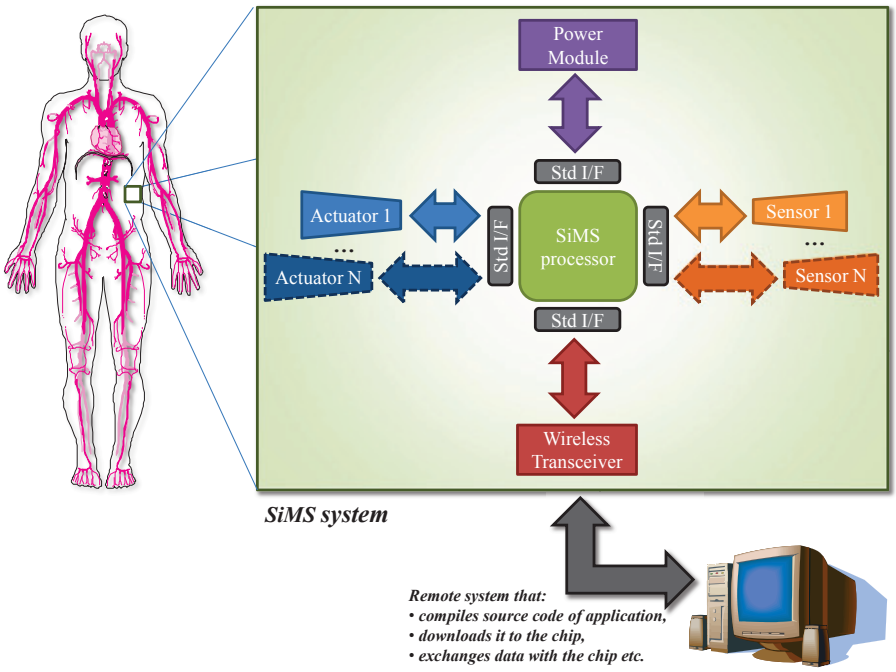


Figure 1.2: *Block diagram of the SiMS concept.*

1.3 Motivation

Operating under the assumption that implants will be the vehicle towards improved, personal healthcare and, in view of the aforementioned transitions, we believe that a *new paradigm in implant design* is required.

In this dissertation we introduce **Smart implantable Medical Systems (SiMS)**. SiMS is a *systematic* approach – a framework – for providing biomedical researchers (and, hopefully, industry) with a toolbox of ready-to-use, highly reliable implant sub-systems and models in order to construct (optimal) implants for various medical applications. The SiMS framework is conceptually illustrated in Figure 1.2; it has to guarantee the following attributes:

- high dependability (reliability, availability, maintainability and safety);
- modular, versatile design for design reuse;
- ultra-low power consumption; and
- miniature size.

Devices built on the SiMS framework will be small, *fine-tuned* implantable devices to the application at hand, yet built of *generic* components. Without requiring redesign, they will be able to *measure* and/or *regulate* one or multiple biomedical parameters simultaneously and communicate with external (out-of-body) computing equipment wirelessly. Given that such devices are directly related to human life, they will be characterized by very high **reliability** and some degree of **autonomy** and **self-awareness** (within extremely demanding **low power** and **size** constraints).

A cornerstone of the SiMS framework is the **SiMS processor** which will be characterized by and will support the aforementioned attributes. After describing the overall SiMS framework, our foremost goal in this dissertation is the specification of the SiMS processor. In effect, we explore optimal processor configurations which are best-suited for serving diverse implant applications (i.e. have universal applicability) while exhibiting low power consumption, low energy expenditure and low area cost.

1.4 Dissertation challenges & contributions

Throughout this dissertation work, we have encountered various challenges of which we were utterly oblivious at the outset. Such challenges are as follows:

- Since the SiMS approach is original, no useful literature or resource base exists: Design specifications, reference designs, established benchmarking platform as well as suitable design tools are unavailable;
- The implant field is ruled by high risks and high profits. Implant companies generally assume a highly conservative (and often secretive) stance towards new product development. Their current state of the art has virtually been inaccessible to us to use for reference. A lot of careful guess-work is required;
- We are attempting to propose a different approach on implant design. As with everything new, this has raised strong reactions from the current status-quo and, conversely, has made us question every new step we made in this unknown ground.

With this dissertation, the following diverse contributions have been delivered:

- Comprehensive survey and analysis of implantable systems revealed crucial trends in the field,

- Conceptualization and setup of a new, top-down design paradigm for implantable systems (SiMS),
- Development of new simulation/evaluation/DSE tools for implant processor design. Development of benchmarking base, and
- Automatic, multiobjective design-space exploration of optimal SiMS processor architectures.

We should make clear that, within the SiMS project, it has *not* been our express goal to propose novel implant applications but, rather, to specify a sound framework upon which many existent (but certainly not all) and, most importantly, new implant applications can be built. SiMS shall guarantee a reduction of development times by providing a solid substrate onto which prior art will be brought together, combined and integrated in the final product. Such prior art will be in the form of Intellectual-Property hardware and software components, all proven, pre-verified and pre-tested according to (inter)national medical-safety regulations. This shall, in turn, guarantee an increase in component- and device-level reliability.

By being fully aware that implantable devices are fruits of a multidisciplinary, combined effort, we also work within the SiMS framework towards a clear separation of partner expertices. That is, we aim at a framework where engineers from different fields provide the system architecture, the sensors and actuators, the power source, the wireless transceiver etc., while medical experts are actively involved in composing, adjusting the final system to the particular patient needs.

1.5 Dissertation organization

The dissertation at hand has been organized in chapters, each handling a different item of study.

In Chapter 2 we present the findings of an extensive *survey* performed on more than 60 different implantable systems, found in the *literature*. To make the analysis manageable, findings have been taxonomized in different categories covering all aspects of modern implantable systems. The chapter concludes by summarizing the most crucial trends observed in the implant domain, thus, providing the scientific background onto which the SiMS project has been based.

In Chapter 3 we discuss at length the *socioeconomic drifts* necessitating the

inception of SiMS. We, then, describe the *SiMS concept* in detail and present the *background information* required for its realization.

Chapter 4 is occupied with defining the *simulation environment* – simulator, benchmarks, input datasets – for our further experiments. The simulator employed is detailed and practical issues are discussed. Original *benchmark programs* are being investigated and the most suitable ones are grouped in a *novel benchmark suite* for implant processors, called **ImpBench**. Proper *input datasets* to these benchmarks are also discussed. With all pieces of the simulation environment finally in place, we conclude the chapter with the *case study* of an instance SiMS-processor application.

In Chapter 5, we offer an in-depth exploratory study on suitable cache organizations and branch-prediction policies for a novel processor for SiMS-based implants. Our standard first-order, optimization goals *performance*, *power consumption* and *energy expenditure* are in this chapter expanded by a third one, *area utilization*.

In Chapter 6, we first introduce **ImpEDE**, a new tool offering *automated*, *multiobjective* DSE (Design-Space Exploration) of optimal SiMS processor microarchitectural configurations. Through ImpEDE, we introduce one more optimization goal – *execution time* – next to performance, power consumption, energy expenditure and area. The need to introduce the notion of "hard deadlines" in program execution have coerced us in developing an updated version of ImpBench (v1.1), reported next. As a last and culminating point in this thesis, we utilize ImpEDE, ImpBench v1.1 and suitable implant applications extracted from the survey in Chapter 2 to offer a number of optimal SiMS-processor solutions.

Last, Chapter 7 provides concluding remarks on the work presented. The chapter summarizes the dissertation, outlines its contributions and proposes future research directions.

2

A survey on microelectronic implants

WHILE at first restricted to the field of pacemakers, biomedical microelectronic implants nowadays boast an expanding number of biomedical applications. These technological innovations have brought about a revolution in existing methods for disease diagnosis and therapy. However, the relatively short lifespan of the implant domain – traditionally subject to tight, demand-driven design policies and bound by economical constraints – has resulted in the absence of a holistic view of the field.

Structured, repeatable methods of implant design are currently sorely missed and previous, precious know-how is currently being wasted and rediscovered. Before any systematic implant-design approach such as the one we take in SiMS is feasible, a careful exploration of the field must take place.

In this chapter we present selected findings of an extensive *survey* performed on more than 60 different implantable systems, found in the literature. To make the analysis manageable, findings have been taxonomized in different categories covering all aspects of modern implantable systems. The chapter concludes by summarizing the most crucial trends observed in the implant domain, thus, providing the scientific background onto which the SiMS project has been based.

2.1 Survey goals

An extensive survey has been performed and serves a **twofold purpose**. Firstly, we create a detailed *implant taxonomy* of a large number of systems. An extensive list of device attributes has been extracted and a subset thereof (pertinent to the rest of this thesis) is presented in this chapter. Information has been collected, organized and is presented in a highly structured manner.

Secondly, we attempt to present a clear picture of the past and present state of things in the field: More than 60 implant cases over the period 1974–2005 have been studied and included in the survey. We also attempt to uncover potential problems along the way, and, through making suggestions for future design, to propose new directions for implant design. Survey results are, thus, extensively *commented* and *annotated*, and analysis is backed by *statistical tests* of the gathered data, where possible.

2.2 Survey scope

For this survey, reported literature dating back to 1974 has been collected but the scarcity of reported study cases in those earlier years has forced us to primarily focus on the more densely populated 12-year period 1994–2005. As we shall see through the course of this analysis, even for this later period, available data is very limited (i.e. *sample size* is small). To make matters worse, collected data is very "noisy" (i.e. *variance* is large), reflecting the aforementioned diverse nature and non-structured approach in implant design. Although both these effects weaken the strength of many of our performed statistical tests, we do include many such tests in the current study for completeness purposes and in the hopes of repeating them in an extended, future study on a larger sample size. Splitting the data into more focused subgroups might solve the variance problem but would result in prohibitively small sample-data sizes which would weaken our analysis.

While the, often, significant data variance is an inherent side-effect of the studied field, we have attempted to collect robust albeit more limited sample data. Robustness stems from the **scope** of this survey which has been restricted to study cases adhering to the following two requirements:

1. **Complete systems:** Stand-alone working devices providing complete functionality are considered; thus, simple implant components such as electronic front-ends, biosensors, readout electronics etc. are excluded.

2. **Microelectronic devices:** Microelectronics- and MEMS-based devices for in-vivo operation are considered; thus, mechanical implants (e.g. artificial limbs), implant packaging, devices for medical studies (e.g. bioassay chips) etc. are excluded.

The rationale behind the above requirements lies in the fact that, first, we are interested in **complete implant solutions** available so far. We wish to investigate functional devices which have been designed with the whole system in mind, under system-wide design considerations. Second, we are especially focusing on the **architecture of the processing and/or controlling cores** (collectively termed PCC's) residing inside such systems since our expertise, interest and prior work chiefly lies in the field of computer architectures. It is, further, our strong belief that the implant architecture is a design aspect to benefit greatly from the recent advances in microtechnology and, due to this fact, also an aspect wherein lies much room for improvement.

Overall, with this study we are primarily interested in presenting how microelectronic implants have developed over the years from a system's perspective. This is not to say that other aspects of implant design such as e.g. device packaging should be overlooked. On the contrary, packing is one of the key elements for chronic implantation of implantable devices these days. However, it pertains to other fields of study like, for instance, materials science and, as such, lies outside the scope of this survey.

It should be stressed that all surveyed systems originate from published literature across various biomedical, microelectronics and other fields of science. That is, almost all devices are academic-level and not commercialized systems, with all that this implies. We are aware that this selection is *biased* in a number of ways: i) surveyed devices may have not reached the *level of optimization* demanded of industrial products, and ii) surveyed devices may not be accurately reflecting the at-the-time commercial state of the art, as devices used for research tend to be more *ambitious* and, in many ways, ahead of their time.

However, we do not expect these biasing effects to have a annulling effect on the findings of this survey for academia and industry alike: As it turns out, the implant domain still is a very limited embedded-systems subdomain whereby the path from a successfully researched new implant to a commercial product is short and straightforward.

What is more – and as will be made evident through the process of this study – most of the studied devices have actually been *implanted* in living animal or human specimens and *in-vivo* evaluated. As a result, functional specifications as well as physical dimensions had to have reached a high level of sophistication[1] before being tested, which is more than most experimental embedded devices can brag in other domains.

[1] See, for instance, the discussion on implant chipset and packaging size in Section 2.6.2. Dimensions are very close to those of existing commercial systems.

In any case, this survey and analysis is a best-effort yet unique attempt at collecting, taxonomizing and characterizing data from a domain traditionally governed by high risks as well as profits; thus, by high secrecy, royalty-protected designs and largely non-disclosed information.

Last but not least, this survey has not been intended as a starting point for discussions on the impact biomedical implants have on societies or their ethical implications thereupon. Without dismissing the *ethical repercussions* arising from the improvement and widespread adoption of such devices, we restrict discussion on strictly technical matters throughout this document.

2.3 Survey structure

Overall survey structure is as follows: Section 2.4 gives a short description of the main components of microelectronic implants and their functionality. In Section 2.5, we discuss at length the imposed classification parameters we have used for organizing the survey information and we present the different types of statistical tests we are going to perform on them. Section 2.6 analyzes the survey findings through the use of illustrated observations and statistical analysis. Overall conclusions are drawn in Section 2.7.

2.4 An implant primer

Since various parameters of microelectronic implants are investigated in this survey, a quick overview of the major implant subsystems is given here. A conceptual illustration of a typical microelectronic implant is depicted in Figure 2.1. As can be seen from the figure, a typical, modern implantable system comprises the following parts:

- **Internal part:** The actual *implant* with peripherals (sensors, actuators, wireless transceiver, power source etc.)
- **External part:** The (optional) external host unit (e.g. PDA) controlling or simply auditing the implant.
- **Communication link:** The (optional) wireless link between the internal and external part for exchange of physiological data and/or implantable-device commands.

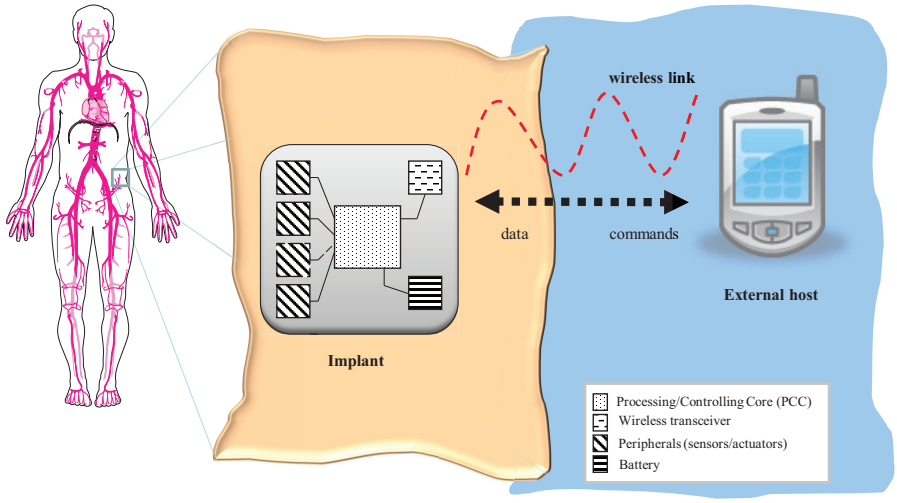


Figure 2.1: *Concept of a typical implantable system.*

The *internal part* consists of combinations of different components, most often, what we have termed a *Processing/Controlling Core* (PCC) lies in the heart of the device coordinating overall operation. Acquisition of physiological data on the part of the implants is usually achieved through appropriate *sensors* whereas intervention to the human body (such as insulin administration, nerve and muscle electrical stimulation, to name a few) is effectuated through *actuators*. One or more sensors and/or actuators collectively termed *peripherals* are included in the implant. Such peripherals realize the physical interface to the body, measuring and/or affecting physiological quantities, respectively.

A common characteristic in many modern implants is their ability to percutaneously accept commands from an external host system (e.g. computer, handheld device) and/or to transmit physiological data outwards, as measured from inside the body over a transcutaneous interface. *Communication* between the internal and external parts is optional yet is increasingly used these days, is commonly achieved wirelessly and can, per case, be unidirectional or bidirectional. For this reason implants also include some kind of wireless transceiver as a peripheral unit to the PCC.

Power inside the implant is provided by a separate peripheral, either an included battery cell or an induction coil receiving EM-power wirelessly from an external coil. Oftentimes, the same coil used for information broadcasting is also used for power transfer through living tissue.

The (also) optional *external part* typically is a (portable) computing unit such as a laptop or a palmtop or, even, a desktop computer. This is under the direct control of the patient, his/her treating physician and/or technician. From the external host, commands can be transmitted to the implant and physiological-data reception can be achieved. The external host often serves also as a data logger, storing large quantities of measured readouts acquired through the implant sensors since implant-native data memory is usually very limited.

2.5 Survey setup

The existence of so many diverse attributes in the studied biomedical implants, makes their classification a non-trivial issue. We have attempted to take into consideration as many of these attributes as possible while, at the same time, keeping complexity manageable. To this end, a two-level hierarchical classification of data has been designed, as illustrated in Figure 2.2.

In the first level, eight major categories have been identified covering core aspects of an implantable system. In the second level, these categories have been further broken down, each into a set of related parameters considered to be pertinent to our study, as outlined in the previous section. For the purposes of this thesis document, we shall limit our analysis to only those categories related to the SiMS project. For an exhaustive analysis of all categories, the interested reader can refer to a complete technical report [133]. Below, we give a detailed description of the selected categories along with their measurement units and the policy followed in filling in the taxonomy tables. The omitted categories have been denoted with an asterisk (‘*’) in Figure 2.2.

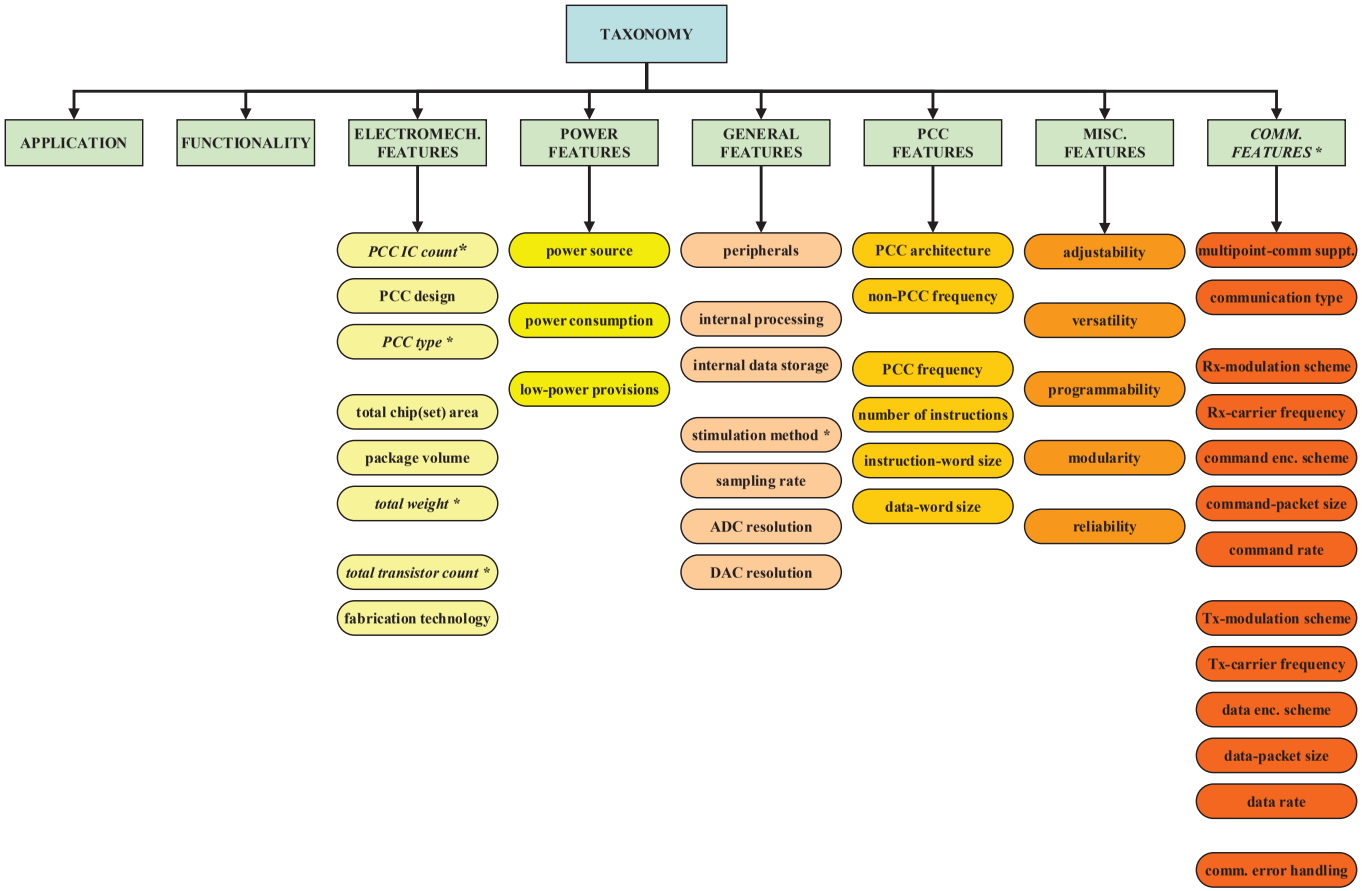


Figure 2.2: *Overview of imposed survey taxonomy.*

2.5.1 Major categories

In detail, seven major categories are described in this thesis, as follows:

1. ***APPLICATION***: The medical problem the specific implantable system is designed to remedy. Intensive-care continuous monitoring, pain therapy, disease diagnosis, restoration of paralyzed limbs are all typical applications.
2. ***FUNCTIONALITY***: The functional principles employed by the implantable system for fulfilling its application purpose. This can typically be sensor-based acquisition of biological data or electrical stimulation of living tissue.
3. ***ELECTROMECHANICAL FEATURES***: The design approach and implementation technology of its microelectronic parts and packaging as well as the mechanical aspects of the implantable system, such as physical size.
4. ***POWER FEATURES***: The power consumption of the implant, the power source used and any implemented special features for low power.
5. ***GENERAL IMPLANT FEATURES***: The most common attributes of microelectronic implants such as supported number and type of sensors and actuators, provision for internal data storage (for data-acquisition systems), supported sampling rate, ADC or DAC resolution and so on.
6. ***PROCESSING/CONTROLLING-CORE FEATURES***: Further and more involved details of the PCC (if present) of the implant. Examples are the core frequency and the instruction- and data-word sizes.
7. ***MISCELLANEOUS IMPLANT FEATURES***: More specific attributes of an implant; for instance, its ability to support various parameter settings, to support different peripheral modules (e.g. sensors), to feature hardware- or software-supported error handling during operation and so on.

The above categories make it obvious that the attempted classification includes almost exclusively attributes of the implanted components of the surveyed systems but not of the external components. External components (as seen in Section 2.4) have diverse features of their own and present their own challenges. However, they are not subject to the tight constraints or requirements their implanted counterparts are. As previously mentioned, the concern in this survey resides mainly on the characteristics of the central PCC that, if present, is implementing (or, at a minimum, supporting) the functionality of the implant.

2.5.2 Minor categories

We have further broken down the major categories, presented above, into 25 parameters[2]. In what follows, each one is analytically presented and its context is explained. It is noted that, for the first two major categories, i.e. *APPLICATION* and *FUNCTIONALITY*, no underlying parameters exist. For the remaining five categories the parameters are as follows:

2.5.2.1 *ELECTROMECHANICAL FEATURES*

i. ***'PCC design'***: The design style (full-custom, semi-custom or based on commercial, off-the-shelf (COTS) components) of the PCC IC(s) in the implant. This field is meaningless if the 'PCC IC count' parameter equals zero and is, in such cases, marked as "non-applicable".

ii. ***'total chip(set) area'***: The area in mm^2 of the PCC chip(s) and/or of the overall implant chipset in case of discrete-component or multi-chip implementation. If more detailed dimensions figures are given (e.g. PCC area and total PCB area), all are reported and distinguished in this field.

iii. ***'package volume'***: The dimensions in mm^3 of the implant packaging.

iv. ***'fabrication technology'***: The fabrication process used for realizing the implant IC(s). When COTS-based or no IC(s) are used in an implant, the fabrication technology used is reported as "non-applicable".

2.5.2.2 *POWER FEATURES*

i. ***'power source'***: The type of power source the implantable device uses.

ii. ***'power consumption'***: The *overall* power consumption of the implant. If the design includes digital (PCC) components, the fraction of the total power consumed by them is included as well (if known) in square brackets next to the overall figure. Also, the operating voltage is reported last, in curly braces. Thus, the form of this field is: `power_total [power_digital] {operating_voltage}`.

iii. ***'low-power provisions'***: The special provisions in hardware and/or software of the implant for achieving low-power state(s) of operation. This

[2] In the technical report [133], the full range of 43 parameters is included.

field does not take into account low-power components used in the design (e.g. low-power amplifiers) since this is almost always the case in such devices. It, rather, pertains to more dynamic or system-level methods of reducing power such as sleep modes of PCC operation, pulse-powering of subsystems etc.

2.5.2.3 *GENERAL IMPLANT FEATURES*

i. ***'peripherals'***: The set of peripherals (stimulating/monitoring electrodes, sensors, actuators etc.) implemented in the implant. In case the implant features a more generic design which theoretically supports a larger or different set of peripherals (thus, a superset of the given instance), they are also reported last, in square brackets.

ii. ***'internal processing'***: The ability of the core (if present) to perform, apart from common signal conditioning operations (like ADC[3]/DAC[4], filtering etc.), further data manipulation, e.g. Fourier transformation, data compression, control feedback, application-specific algorithms etc.

iii. ***'internal data-storage capability'***: The ability of the implant to store data internally, i.e. in some internal memory block. Such data can be physiological recordings, stimulation control settings etc. If the data-memory type and size utilized are available, they are also reported in this field.

iv. ***'sampling rate'***: The sampling rate in `Hz` supported by the implanted system if/when biological-data acquisition takes place.

v. ***'ADC resolution'***: The resolution in `bits` of the included ADC(s) in an implant (if present).

vi. ***'DAC resolution'***: The resolution in `bits` of the included DAC(s) in an implant (if present).

2.5.2.4 *PROCESSING/CONTROLLING-CORE FEATURES*

i. ***'PCC architecture'***: The structural nature of the implant PCC (if present). This can be a custom or commercial μC or μP, a simple Finite-State machine (FSM), a hardware counter or other.

[3] ADC: Analog-to-Digital Converter
[4] DAC: Digital-to-Analog Converter

ii. ***'PCC frequency'***: The PCC(s) clock frequency in `MHz` (if present).

iii. ***'non-PCC timing'***: If a PCC is present in the design, this field is non-applicable. Otherwise, this field holds the (highest) running frequency in `kHz` of the overall implant design, e.g. for some included Switched-Capacitor (SC) circuitry.

iv. ***'number of instructions'***: The number of available instructions featured by the PCC(s). If the maximum number of instructions (as specified by e.g. the opcode bits) is known, then it is noted in curly braces right next to the first number (if known). If the two numbers coincide, there is no maximum number included.

v. ***'instruction-word size'***: The core instruction size in `bits` (if present).

vi. ***'data-word size'***: The core data size in `bits` (if present). Since many designs are highly customized, it may happen that this figure does not coincide with the instruction-word size. In cases where internal memory is included in the implant, this size is typically equal to the memory width.

2.5.2.5 *MISCELLANEOUS IMPLANT FEATURES*

i. ***'adjustability'***: The capacity of the implant to accommodate diverse operational settings (e.g. sample rate, filter bandwidth, amplifier gain, stimulus-pulse duration and amplitude, sensor sensitivity etc.) of its peripherals (e.g. sensors, actuators).

ii. ***'versatility'***: The ability of the implant to serve in different operational roles by being able to drive different peripheral modules.

iii. ***'programmability'***: Characterizes implants featuring a program (and data) memory with specific downloaded code for achieving their functionality. This parameter does not refer to temporary data storage as e.g. in the case of FSM's where data are kept in registers for controlling the FSM state transitions. The parameter pertains to the ability of an implant to execute different source codes rather than operating based on some hardwired or hard-coded function. If available, the type and size of the program memory is also reported in this column. Also, if the PCC can be in-system reprogrammed, that is, different source codes can be downloaded and executed after implantation, this trait is also reported.

iv. ***'modularity'***: The design nature of the PCC(s) (if present). It describes *generically* designed cores that can inherently (i.e. without modifications) support many different biomedical applications by allowing the plugging-in of a large (infinite, theoretically speaking) set of different peripherals. This field also refers to any other feature that extends the (re)usability of part or whole of a specific design to other designs.

v. ***'reliability'***: This trait includes all provisions made in an implant design (either in hardware or in software) for incorporating error-handling functionality. This may range from simple error-detection to error-correction or -recovery techniques. It also entails fault-tolerant design, built-in diagnostic systems, design for testability etc. Mechanical aspects such as packaging, assembly and materials are not considered.

2.5.3 Discussion

In the above, we have laid out a rather extensive list of taxonomy parameters against which we have queried the collected implant designs for data. The accommodated parameters, while focusing on the PCC component(s) of the implants, address other crucial aspects of implant designs, as well. Such aspects include low power and small feature size as well as value-creating enhancements; namely, wider versatility, improved (multi)functionality and built-in reliability, to name a few.

Except for offering a taxonomized and highly structured review of related art, these parameters serve a second, more crucial role. They are essentially used as the statistical *variables* encoding all gathered information in a form suitable for statistical analysis. In this analysis, these variables have been used in various illustrations and statistical tests for extracting educated conclusions in the field, as will be presented in the following sections.

Although there is an almost one-to-one relation between the taxonomy data and the variables, some differences exist. For instance, taxonomized data belonging to one parameter (e.g. 'power consumption') had to be split in more than one actual variables (e.g. 'standby power', 'average power', 'transmission power', 'active power' and 'operating voltage') to allow for accurate statistical tests to be performed.

A final word on terminology: In this dissertation, the terms "(taxonomy) parameters" and "(statistical) variables", although slightly different, shall henceforth be used interchangeably, referring to the same data quantities seen from a different perspective, unless otherwise stated. The terms "PCC" and "core"

also refer to the same implant component. Last, commonly met components in implantable systems such as sensors, actuators, electrodes and even wireless-communication modules shall also be collectively referred to as "peripheral units", "peripheral modules" or, simply, "peripherals".

2.5.4 Statistical tests

Through the course of the survey analysis, numerous research questions have been raised, the answers to which have been formulated into five statistical tests. In order to familiarize the reader with the following analysis, five representative research-question instances, along with their corresponding tests, are presented next.

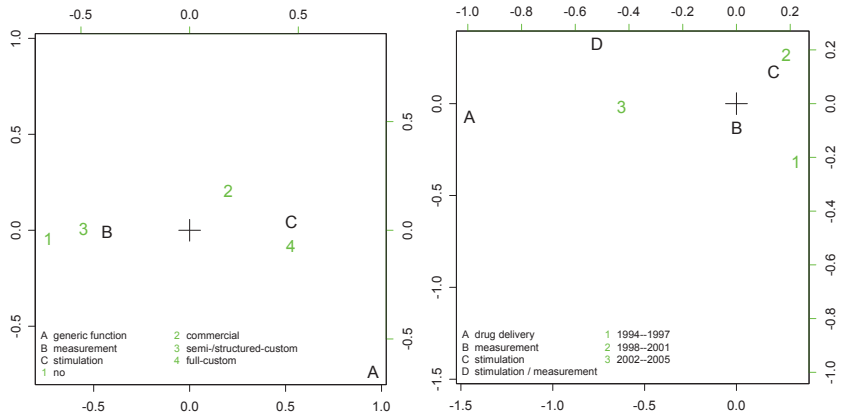
2.5.4.1 Testing independence of two categorical variables

> Question: *"Does implant **functionality** have an effect on the **PCC design** employed?"*

This question translates to the equivalent question whether there is a strong correlation between the two variables 'functionality' and 'PCC design'. Both the dependent (or response) variable 'functionality' and the independent (or explanatory) variable 'publication year' are *categorical*. In effect, a suitable statistical test to employ for testing the hypothesis between two categorical variables is a *chi-square test*.

In this case, the test reveals a significant correlation between the variables at the .05 level ($p = 0.0235$). Since the variables contain multiple categories, we would like to explore further the relation between the various categories. We could try to investigate this through a scatterplot but since both our variables are categorical (i.e. non-numerical), we have plotted a Correspondence-Analysis (CA) plot, as shown in Figure 2.3a. The plot does indicate there is a correlation between the two variables and we further identify relations between their categories; for instance, most stimulation implants have been developed as full-custom processes.

There are, however, other cases in this study that the chi-square test returns non-significance, meaning that the null hypothesis cannot be rejected (i.e. implant functionality does not change over time). Unfortunately, this is often due to the large standard error of the limited sample data. Even in those cases, we would like to get a feeling of what the relation between the categories of the



(a) *CA plot of PCC design vs. functionality.*

(b) *CA plot of functionality vs. publication year.*

Figure 2.3: *CA plots between two categorical variables.*

two variables is. In such cases, a CA can also be used.

2.5.4.2 Testing whether categorical variables change over time

> Question: *"Do the relative percentages of implant-**functionality** categories change significantly over **time**?"*

This question translates to the equivalent question whether there is a strong correlation between the two variables 'functionality' and 'publication year'. In this case, the response variable 'functionality' is *categorical* while the explanatory variable 'publication year' is *scale*. However, as will be explained later in the analysis, publication years can be grouped into 3 time periods (i.e. categories), effectively allowing for treating 'publication year' as a categorical variable, too. Therefore, for testing the hypothesis of the above research question, a *chi-square test* and assorted CA plot can be used in this case, too. In fact, in this case the test does not reveal a significant correlation (p = 0.4240) between the two variables but the CA plot (Figure 2.3b) can reveal some interesting trends.

2.5.4.3 Exploring the relation of a scale variable over time

> Question: *"How does implant **power consumption (peak active)** change over **time**?"*

We wish to explore how the scale variable 'power consumption (peak active)' changes over time which is represented by the scale variable 'publication year'. This type of question can be answered by *simple regression analysis*. Both linear and polynomial regression lines have been fitted through least-squares (LS), least-median-of-squares (LMS) and least-trimmed-squares (LTS) estimators, as seen in Figure 2.4. A simple (non-parametric) smoother line has also been plotted. As the smoother line also hints, a quadratic regression line based on LS has been fitted to the data, as shown in Figure 2.5. Confidence-Interval (CI) bands of 95% have also been plotted.

2.5.4.4 Exploring the relation of a scale variable over time over groups

> Question: *"How does **power consumption (peak active)** change over **time** for different implant **functionalities**?"*

Through asking this question we wish to investigate whether implants from the two major functionalities (measurement and stimulation) exhibit different power-consumption trends over time. In statistics jargon, we wish to determine whether 'functionality' affects 'power consumption (peak active)', controlling for 'publication year'. For this kind of questions, a so-called *dummy-variable regression model* is created accommodating also interactions between the two categories (or factors) of dummy-variable 'functionality' and the independent scale variable 'publication year'.

For this question instance, let us revisit Figure 2.5 and highlight with different colors the two categories of the dummy variable; see Figure 2.6a. On the scatterplot, (non-parametric) smooth curves have also been drawn to assist exploration. Quadratic regression lines have been fitted on both stimulation and measurement points and are illustrated in Figure 2.6b. Although the shape of the two fitted curves is not identical, we wish to test whether the observed difference between them is statistically significant. Therefore, we append to our analysis a Log-Likelihood ratio (G) test which yields non-significance ($G = 1.0080$, $p = 0.7993$). For the particular question, this outcome means that the two implant functionalities do not differ significantly in their power-consumption trends over time to merit separate handling.

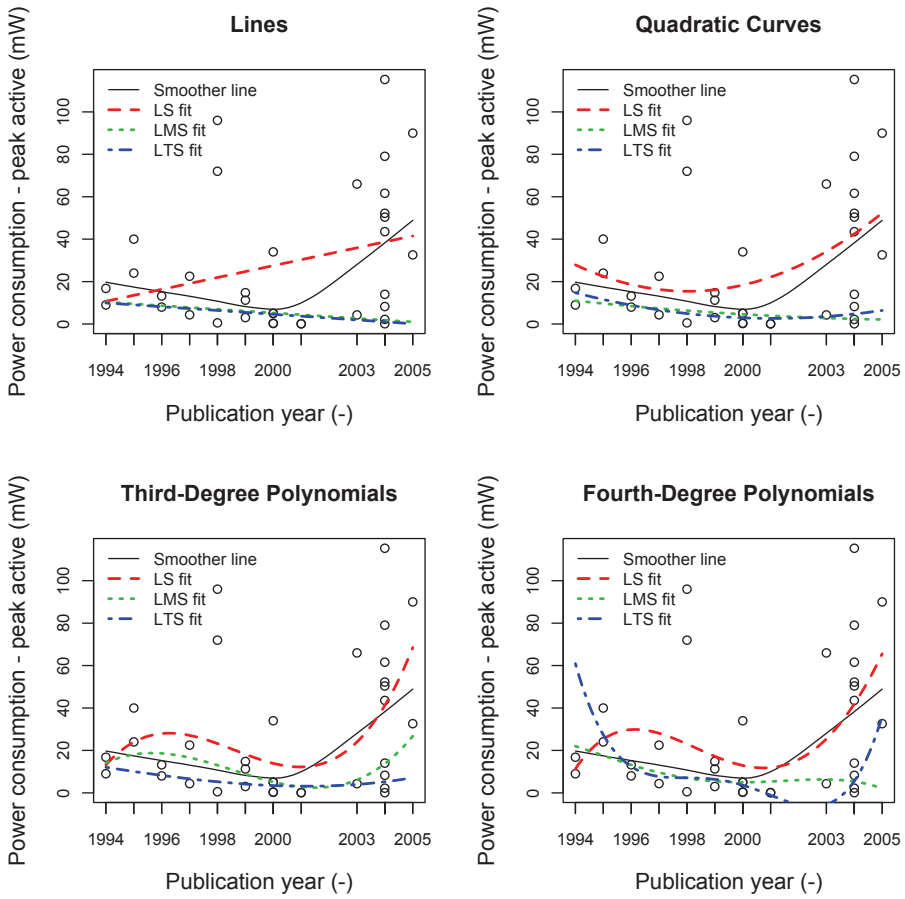


Figure 2.4: *Various regression lines fitting the power-consumption (peak active) data over the years.*

2.5.4.5 Comparing a scale variable over groups

> Question: *"Does the chosen implant **PCC design** has an impact on its **power consumption (peak active)**?"*

We wish to perform a "within-groups" comparison of a given categorical variable (here, 'PCC design') with respect to a scale variable (here, 'power consumption (peak active)'). It is interesting to see whether power consumption is generally affected by implant PCC design. Exploration has been performed numerically, through a *Kruskal-Wallis rank sums test*, and visually, through *boxplots*.

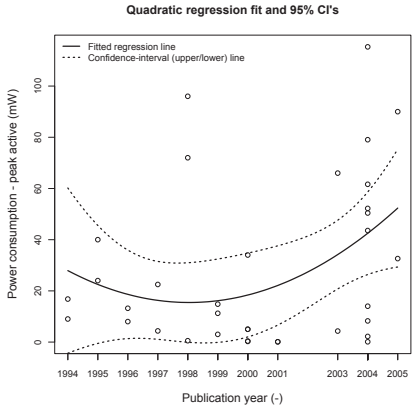
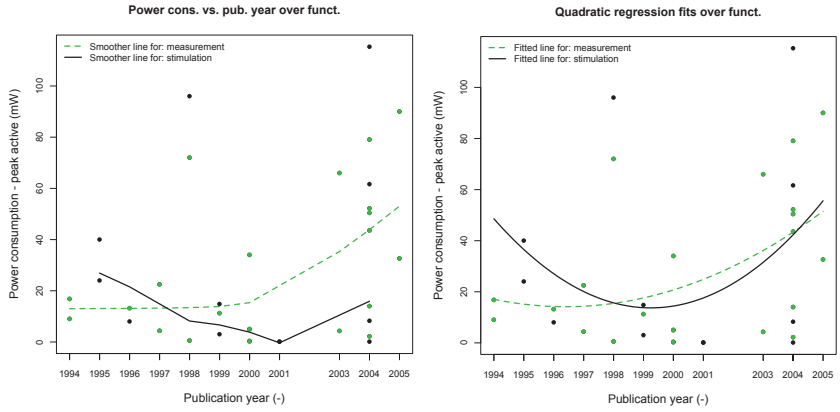


Figure 2.5: *Implant power-consumption (peak active) trends over the years.*



(a) *Smoother lines.* ***(b)*** *Fitted quadratic lines.*

Figure 2.6: *Power-consumption trends over the years for different implant functionalities.*

The boxplots in Figure 2.7 indeed reveal different median power profiles for implants employing different PCC designs. For instance, full-custom implants consume, on average, more power than ones built of off-the-shelf components. Of course, implants with no PCC at all consume the lowest power overall. Even though such differences are easily visible in the figure, we wish to use some suitable statistic to verify them. Since the comparison is "within-groups" and we have a combination of a scale and a categorical variable, a Kruskal-Wallis (KW) rank sums test appears to be the most suitable choice. For this par-

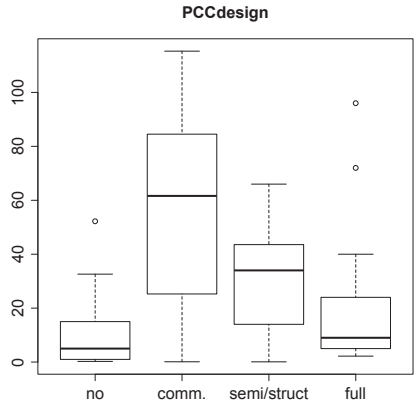


Figure 2.7: *Boxplot of implant power consumption (peak active) with respect to PCC design.*

ticular research question, a KW test returns non-significance (X^2 = 3.9675, p = 0.2650).

2.6 Survey results

In the following sections, survey findings will be presented and analyzed in an organized fashion. An important issue to keep in mind is that in the current study we have attempted to cover a wide as well as representative range of existing implantable devices (either for research or for commercial purposes). Nonetheless, we are aware that the field has not been covered in a perfectly homogenous manner. As a result, the statistics generated hereafter are at times skewed which will be diligently discussed throughout. Even so, they currently constitute the best possible (and, to our knowledge, the only) study of the present state of implants.

2.6.1 Implant applications & functionality

It is highly didactic to, first, take a close look at the application spectrum microelectronic implants are covering these days and the functionalities they are employed for. Thus, the categories *APPLICATION* and *FUNCTIONALITY* are investigated first. The *APPLICATION* column, reveals the wide spreading of implantable devices across diverse medical fields. As will be shown below, these devices generally serve two distinct functionalities: *diagnosis* & *therapy*.

Modern implants display a largely (micro)electronic nature which has been an excellent substrate for developing electricity-based measurements. What is more, their miniature size has enabled researchers to build minimally invasive systems for the first time. By deploying such systems, measured-signal distortions due to e.g. electrodes penetrating the skin (conductive medium) have been averted, resulting in precise, low-noise measurements [12].

These traits have encouraged the development of many implants for *diagnostic* purposes. Commonly encountered ones - and amply met in this study - are in-vivo *electrocardiography* (ECG) [38], *electromyography* (EMG) [73] and *electrooculography* (EOG) [109] while an increasingly popular of applications involves *electroneurography* (ENG) [4], i.e. the recording of neural signals (biopotentials) from the central (CNS) or peripheral (PNS) nervous system. Typical cases of ECG-enabled implants are implantable *pacemakers*, which deliver cardiac pacing, and implantable *intra-cardiac defibrillators* (ICD's), which monitor the cardiac rhythm and attempt to prevent or to counter any spontaneous cardiac arrhythmias (antitachycardia pacing, cardioversion, defibrillation) in an automated, closed-loop fashion.

Recording of neural activity (i.e. ENG), on the other hand, has long been pursued by physiologists as a means to understanding the operation of individual neurons, to deciphering the organization and signal-processing techniques of biological neural networks and to controlling a variety of prosthetic devices. Implantable devices have contributed to rapid developments in the field as they have permitted physicians and engineers the unprecedented ability to perform the measurement, conditioning and storage of such highly sensitive signals (range can vary from $10\ \mu V$ to $100\ mV$) in-vivo. To give a feeling of the popularity of this class of applications, 13 in a total of 60 entries (about 22%) have been encountered with the actual percentage anticipated to be much higher.

Apart from the above, a plethora of other in-vivo measurements has been achieved through implants: *body temperature* [152], *intracranial pressure* (ICP) [45] for preventing brain diseases (especially in post-operation, head-trauma patients), *pH* [107], *blood pressure* (BP) [123], *gastric pressure* [140], *renal-sympathetic-nerve activity* RSNA [38], *cardiac output* for monitoring the performance of an artificial heart valve [121] and *tissue bio-impedance* [94] have also been implemented in implantable systems. Also, *graft monitoring* (through pulse oximetry) following organ-transplantation surgery [39] and intra-articular mechanical stress such as *tibial-force monitoring* inside titanium implants [32] are less widespread, yet important contributions to the field.

A physiological parameter that has received special attention, and is also present in this study, is *glucose concentration* in the blood. This is a hot topic these days given the large (and steadily increasing) number of diabetic patients worldwide. High glucose levels (hyperglycemia) in the blood stimulate the pancreas for releasing insulin to lower glucose concentration. Since in diabetic patients the pancreas cannot produce insulin, an "artificial pancreas" is required, virtually a closed-loop control system which samples the glucose levels in the blood stream and releases insulin as needed. Glucose-level-sensing implants constitute only half of this control circle, the other half being drawn by implantable micropumps for administering insulin. Even though the design of an actual, chronic artificial pancreas has not been achieved yet, due to pending unsolved issues, the glucose-sensing implants have constantly increased in numbers and improved over the years. Examples of such attempts are the work of Shults et al. [123] and Beach et al. [13]. Lastly, regarding diagnostic-purpose implants in general, it is important to mention that a significant number of modern devices is capable of achieving continuous, real-time measurements over long periods of time and at a reasonably low power budget (as will be shown in the following subsections). Of course, not all diagnostic scenarios require continuous monitoring of physiological parameters (e.g blood pressure), but rather a periodic sampling.

Diagnostics aside, electricity-based *therapy* has equally, if not further, benefited from advances in microelectronics technology and has been enriched with many research efforts over the years with many of them turning into successful commercial products, too. The first and most prominent application of electricity-based therapy has been cardiac pacing and defibrillation for treating cases of cardiac arrhythmia. Ex-vivo pacing techniques have gradually given way to in-vivo, implantable ones. The pacing implants are the above mentioned pacemakers which are also the first microelectronic devices to have ever been successfully implanted into a human being. The success of these two types of implants has been tremendous. Indicative of the penetration and impact pacemakers have achieved is the fact that, in the U.S. alone, a total number of 180,000 implantable pacemakers have been registered for the year 2005 (source: American Heart Association [35]). The implantable pacemaker, apart from saving lives, has acted as a catalyst on the general public closed-mindedness against biomedical implants.

These devices are principally the same today as the first pacemaker that was implanted 50 or so years ago. Since then, they have come a long way and now - in their fourth generation - encompass a multitude of features and have a battery-life expectancy of almost 10 years. In this study, 3 such sys-

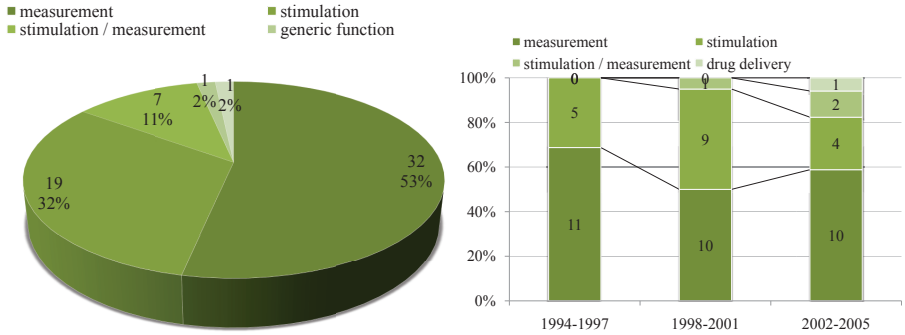
tems have been included: Berkman and Prak [15] present work onto which the *Cordis Sequicor II* and *Gemini* commercial pacemakers have been based. Stotts et al. [131] improve that work, while Harrigal and Walters [56] present results on another commercial pacemaker, the *Kelvin II*. All 3 of them are rate-responsive, dual-chamber (atrial-ventricular) pacemakers with the one from Stotts et al. also making provisions for monitoring of ECG and temperature[5].

The implantable pacemakers and defibrillators have paved the way for further therapeutic uses of electrical stimulation in biomedical applications. One large family of applications, as the study also indicates, is the restoration of various body functions through *functional electrical stimulation* (FES). Muscle stimulation has been attempted for restoration of paralyzed-limb movement, hand grasp [127], micturition and bladder control [119], eyesight [120], vocal [57], hearing [161] and other pathoses. Muscle stimulation is achieved directly (through stimulation of denervated, degenerated muscles, DDM's) or - more commonly - indirectly (through stimulation of nerves, i.e. functional neuromuscular stimulation, FNS). An important and relatively fresh field of research which has largely benefited from implant technology is chronic *Deep-Brain Stimulation* (DBS). This kind of stimulation has yielded phenomenal results for patients suffering from Parkinson's Disease (PD) and is most promising for treating epilepsy and psychiatric diseases [51]. Stimulation has also been employed by *chronic-pain* patients for interrupting nerve (pain) signals from the spinal cord to the brain (e.g. Spinal-Cord Stimulation, SCS [97]). Moreover, it has found use as a method to *regenerate damaged nerve tissue* [49].

Indicative of the impact of electrical stimulation is the large number of such devices present in the study. As Figure 2.8a illustrates, in total 26 out of 60 ($\sim$43%) studied systems implement some sort of electrical stimulation. The Figure also reveals that, together with monitoring implants, they are the two most commonly encountered implant functionalities. As stated before, this makes perfect sense given the microelectronic nature of the studied devices: It constitutes an excellent vehicle for applications where measurement of (bio)electrical signals or tissue stimulation through electrical pulses occurs.

Figure 2.8b illustrates the data from a different angle. It plots the relative frequencies of each one of the different 'Functionality' categories over time. The available data has been grouped into 3 consecutive time periods. It should be noted that, in this as well as similar *trend* plots to follow, trends over the

[5]The above mentioned pacemakers do not reflect the plethora of existing implantable pacemakers and defibrillators. Yet, they were the only ones providing some useable information regarding their design specifics, their power consumption and so on.



(a) *Overall functionality distribution.* ***(b)*** *Functionality trends over the years.*

	1994-1997		**1998-2001**		**2002-2005**	
	Pi	$\pm\sigma$	Pi	$\pm\sigma$	Pi	$\pm\sigma$
measurement	0.69	0.116	0.50	0.112	0.59	0.119
stimulation	0.31	0.116	0.45	0.111	0.24	0.103
stim./meas.	0.00	0.000	0.05	0.049	0.12	0.078
drug delivery	0.00	0.000	0.00	0.000	0.06	0.057
#N	**16**		**20**		**17**	

(c) *Probability estimates (p_i) and their standard errors SE (σ_{p_i}).*

Figure 2.8: *Implant functionality.*

time period 1974–1993 have been omitted due to the prohibitively low sample size, as discussed in the introduction. A table accompanying the trend plot with corresponding descriptive statistics[6] per each time period is also plotted in Figure 2.8c.

In Figure 2.8b, trends over the years indicate that measurement (i.e. monitoring) systems to be the most popular over time, as also seen in Figure 2.8a[7]. One can also discern the appearance of more complex systems over the more recent years. Namely, implants have appeared which combine both stimula-

[6]The 'functionality' variable is categorical with data following a multinomial distribution; each category appears (or not) a number of times in each time period. The probability estimate of appearance p_i is reported in the table along with its standard error σ_{p_i}.

[7]In the time period 1998–2001, a relative increase of stimulating implants can also be observed which causes monitoring implants to somewhat recede. As will be discussed in the following sections, this shift is a small sampling "artifact" introduced due to a relatively large number of reported ocular-restoration (i.e. stimulating) implants by a single or related research groups in the particular time period.

tion and measurement capabilities. That is, they offer *closed-loop* functionality by automatically adjusting one property based on another. To exemplify, implants by Harrigal et al. [57] and Au-Yeung et al. [7] induce so-called *rate-responsive* cardiac pacing by utilizing in-vivo ECG measurements. Besides, the implant designed by Smith et al. [127] implemented restoration of hand function (through muscle stimulation) for persons with tetraplegia at the C6 level (of the spinal column). It offers automatic control and sensing of a joint angle-transducer implanted in the radio-carpal joint of the wrist, with the wrist position used as the command control source. Another type of sophisticated implants that has appeared in the most recent study time-period is drug-delivery implants. Such implants also operate in a closed-loop fashion by releasing regulated amounts of stored drugs into the body based on real-time, in-vivo measurements they perform. The single such case encountered in this study is an implant by Cross et al. [25] which allows for automated oestrus-cycle control and data telemetry in dairy cows.

Question 2.1 *Do the relative percentages of implant-*functionality *categories change significantly over time?*

Figure 2.8b has revealed a noticeable shift in implant functionalities over the years. Yet, it is interesting to also know how "statistically certain" this observation of ours is. The above question translates to the equivalent question whether there is a strong correlation between the two categorical variables 'functionality' and 'publication year' (grouped per time period). We test this hypothesis through a *chi-square test* (as introduced in Section 2.5.4.2). The value of chi-squared is small ($X^2 = 5.990$) and the chi-square test is not significant ($p = 0.4240$), meaning that the null hypothesis (implant functionality does not change over time) cannot be rejected. Weak as the correlation is, we would still like to get a feeling of what it looks like. We have plotted a CA plot, as shown in Figure 2.9. The plot does indicate there is a relation between the two variables: Measurement implants are more proximal to the first time period, stimulation implants are closer to the second period while the more sophisticated, drug-delivery implants are right on top of the most recent time period; thus, the CA plot hints in favor of our initial observations. Last, based on the displayed distances, implants combining stimulation and measurement capabilities are becoming more popular over time.

Increasingly more systems with impressive functionalities are appearing every day, such as bioassay chips, DNA-diagnosis chips or the micropump (mentioned above), which is also used for controlled, *in-vivo drug delivery*. Yet, the functionality and focus in such systems heavily relies on the design of suitable microstructures which are based on MEMS – more commonly known as BioMEMS. Thus, these devices depart from the definition of "system" as given previously – Section 2.4 – and invite different disciplines such as chemical and material sciences. As such, they are out of the scope of this survey.

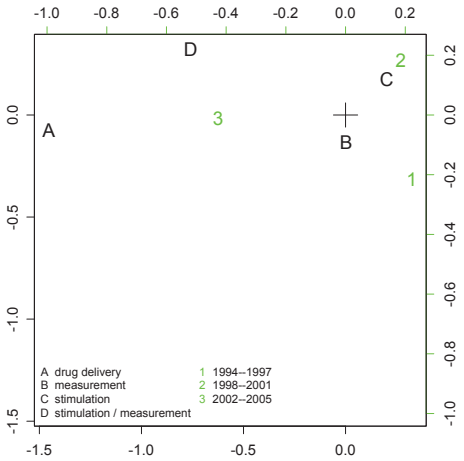


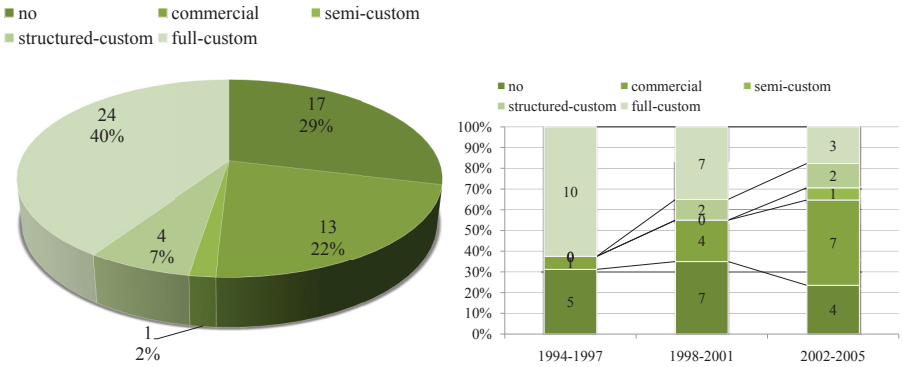
Figure 2.9: *CA plot of functionality vs. publication year.*

As a last remark, we should note that – with the exception of the pacemaker and few other systems – most of the above discussed families of implantable systems have not widely penetrated the biomedical market. Even devices that are fully miniaturized and properly packaged for implantation seem to have remained simple prototypes in a lab bench. The reasons for this situation are varied – for instance, the unsatisfactory chronic in-vivo behavior of packaging materials in the commercialized BION^{TM}implants [138]) – but are, too, outside the scope of the current survey.

2.6.2 Electromechanical features

The survey reveals that implantable devices are most commonly built around some sort of central-processing and/or -controlling unit – the above mentioned PCC. As will be explained shortly, implemented PCCs most often are *full-custom* (ASIC) designs incorporating *mixed-signal* circuitry. The dominance of full-custom design can be readily justified by the fact that most stringent design constraints need to be squeezed in as little circuit area as possible, thus, not allowing the luxury of a design based on discrete components. Power and area are attributes that readily benefit from full-custom design since implementations tend to display lower power requirements and take up less space compared to ones based on COTS components.

Figure 2.10a indeed shows that, in an overall, 42 out of 59 (70%, one unspecified) studied implants include some kind of PCC. The *'PCC design'* attribute



(a) *Overall PCC-design distribution.* ***(b)*** *PCC-design trends over the years.*

	1994-1997		1998-2001		2002-2005	
	Pi	$\pm\sigma$	Pi	$\pm\sigma$	Pi	$\pm\sigma$
no	0.31	0.116	0.35	0.107	0.24	0.103
commercial	0.06	0.061	0.20	0.089	0.41	0.119
semi-custom	0.00	0.000	0.00	0.000	0.06	0.057
struct.-custom	0.00	0.000	0.10	0.067	0.12	0.078
full-custom	0.63	0.121	0.35	0.107	0.18	0.092
#N	**16**		**20**		**17**	

(c) *Probability estimates (p_i) and their standard errors SE (σ_{p_i}).*

Figure 2.10: *Implant PCC design.*

captures the PCC design practice employed for these 42 implants. As can be seen from the pie chart, of the 42 devices: 13 are commercial (22%), 1 is semi-custom (2%), 4 are structured-custom (7%) and 24 are full-custom (40%) designs. Full-custom devices clearly dominate the field. However, should we study the design styles used over different time periods, an interesting trend emerges. Figure 2.10b provides a distribution over the last 12 years. It reveals an initial domination of full-custom designs which gradually makes way for PCCs based on commercial components. This phenomenon is counterintuitive since custom design in a field usually follows rapid prototyping (through use of COTS components) as knowledge of the field grows. The primary reasons for this inverse process in the case of implants are anticipated to be the following:

1. **Development costs**: Early implants had to be as small as possible to fit inside the body. However, current technology miniaturization (CMOS, in

particular) has led to such small form factors that modern implants can be built more economically with discrete components, and still retain a size suitable for implantation.

2. **Development times**: A large fraction of the studied implants is built as prototypes, for proof-of-concept purposes or for initial in-vivo tests in animals. Thus, for the sake of rapid prototyping, COTS components have been increasingly favored in many a case.

3. **Testing/approval times**: By using COTS components to build PCCs, there exists the added benefit of working with pretested, pre-verified, proven cores. System integration is faster and more relaxed in terms of medical approval[8] and testing.

4. **Enhanced functionality**: Implants built around commercial PCCs are – generally speaking – more flexible designs than full-custom ones, allowing for extra functionality (than the originally intended one). This allows more trial and error, especially in experimental setups. The flexibility issue will be further discussed in Section 2.6.6.

While systems with a COTS core have been expanding at the expense of those with a full-custom core or with no core whatsoever, nonetheless, the careful observer can anticipate an increase in coreless systems in the middle period 1998–2001. As discussed in the previous section, this is a sort of biasing "artifact" in the data: By closely studying the collected data for the particular time period, we notice a lot of highly application-specific implants. Namely, a number of intracranial (e.g. ICP monitoring [45]) and intraocular (blurred-cornea treatment [113]) devices with extremely stringent power and size constraints have been reported. As a result, extremely stripped-down implants have been built, with hardwired control and, thus, lacking a PCC. A total of 8 such devices are included in a sample size of 20 device cases for the given time period, which affects statistical analysis significantly. Even so, it is interesting to note that the bias (henceforth called "coreless bias") disappears in the third time period 2002–2005. This supports our initial analysis on the dominance of systems with commercial PCCs. It is also indicative of a potential shift in the targeted implant applications.

[8] Unlike the general case, we are aware that, in this survey, implant medical approval may have been a secondary goal since most studied systems are research prototypes rather than fully commercialized products.

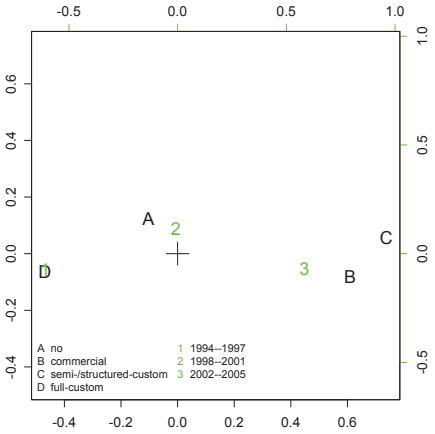


Figure 2.11: *CA plot of PCC design vs. publication year.*

Question 2.2 *Do the relative percentages of implant-*PCC-design *categories change significantly over time?*

This question translates to the equivalent question whether there is a significant relation between the two categorical variables 'PCC design' and 'publication year'. The statistical test to employ is chi-square. The value of chi-squared is $X^2 = 12.1385$ and the test is not significant ($p = 0.0590$), meaning that the null hypothesis is marginally not rejected. In order to qualitatively explore the relation between the variables further, a CA plot has been plotted again in this case, as shown in Figure 2.11. The plot does indicate there is a relation between the two variables: Full-custom designs clearly dominate the earlier years 1994–1997 while COTS-based designs are the most popular choice over the latest period 2002–2005. Coreless designs dominate the middle period 1997–2001 but this is partly due to the coreless (sampling) bias in the same period. What is also interesting is the appearance of semi-/structured-custom designs over the last two periods. This observation further supports the arguments that: i) rapid prototyping in (experimental) implants is becoming a serious driving factor for designers, and ii) (re)design flexibility is at least as important a design parameter as ASIC miniature size and low power consumption. It appears, thus, designers are willing to sacrifice some resources in order to get results faster and to be able to modify a design multiple times for refinement, in-vivo fine-tuning and other purposes.

For all (PCC-enabled) *non-COTS-based* implants shown in Figure 2.10a, the *'fabrication technology'* used is almost with no exception either CMOS (27 out of 55, 49%, 5 unspecified) or BiCMOS (11%). The reason is rather obvious: CMOS-technology features that are highly suited for biomedical-implant applications; namely, very high integration, low power consumption and large noise margins. Bipolar CMOS (BiCMOS) has been mostly used by researchers to construct mixed-signal devices. In addition to the standard CMOS attributes, BiCMOS technology further offers large current-driving capabilities (very important, for instance, for implantable stimulators) as well as intrinsic protection

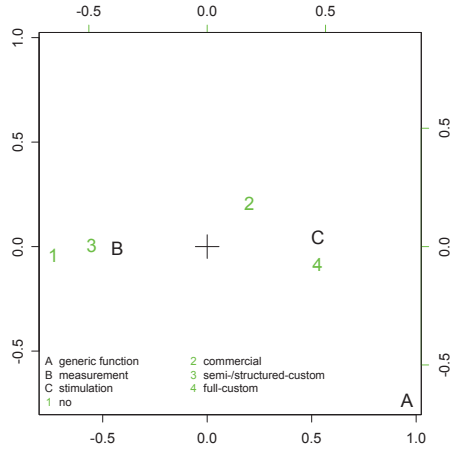


Figure 2.12: *CA plot of PCC design vs. functionality.*

Question 2.3 *Does implant* functionality *have an effect on the* PCC design *employed?*

The relation between the categorical variables 'PCCdesign' and 'functionality' is investigated through a chi-square test (as introduced in Section 2.5.4.1). The relation is significant at the .05 level ($X^2 = 14.6091$, $p = 0.0235$), meaning that the intended implant functionality does indeed affect the design of its PCC. To explore this relation further, a CA plot is plotted, as shown in Figure 2.12. The plot reveals that measurement implants are mostly based on semi-/structured-custom PCC designs or completely coreless designs whereas stimulation implant are most often based on full-custom PCCs and less often on commercial components. Measurement implants are, generally speaking, more passive devices than stimulation ones, meaning that some hardwired (thus, coreless) solution for collecting physiological data will in many cases be sufficient. Stimulation implants, on the other hand, usually require more active involvement such as decoding of externally received stimulation commands, generation or reproduction of stimulation patterns etc.; thus the core-enabled solutions. Also, modern stimulation implants include some measurement capabilities for performing closed-loop stimulation, e.g. rate-responsive pacemakers. This need for more sophisticated control can further explain the stimulation-implants' affinity to high-performance (core-based) designs. However, given the aforementioned trends of implant functionality over time (Research Question 2.1) and PCC design over time (Research Question 2.2), we cannot rule out the chance that the currently observed correlation is coincidental, with both variables only dependent on time.

(e.g. integrated Zener diodes) of the implant electronics against high voltage and current surges.

While the dynamic power consumption of CMOS IC's is excellent, their static power component starts dominating the overall power budget the further fabrication technology (i.e. λ factor) shrinks. This is, admittedly, a potential problem for future implant designs operating in low-power modes over prolonged periods of time (thus, impacting static power). Nevertheless, it is expected to

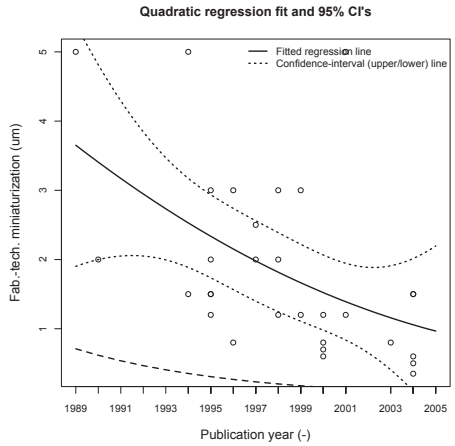


Figure 2.13: *Implant and mainstream microelectronics fabrication-technology trends over the years.*

manifest more slowly as implantable systems follow mainstream market trends (e.g. high-performance computers, portables etc.) with some delay, as the next Research Question reveals.

Question 2.4 *How does implant* fabrication technology *change over time?*

We wish to explore how the scale variable 'fabrication-technology miniaturization' changes over time. Time is represented by the scale variable 'publication year' and the question can be answered by *simple regression analysis*, as introduced in Section 2.5.4.3. Given (i) the observation that the smoother line generally resembles a quadratic curve and (ii) the known fact that transistor sizes are *monotonously* shrinking over time, a least-squares, quadratic regression line has finally been fitted on the data, as shown in Figure 2.13. Confidence-Interval (CI) bands of 95% have also been plotted. Also plotted, as a reference line, are the fabrication-technology trends of the mainstream-market. This line has been calculated based on a typical 13% annual drop, as extracted from the ITRS'04 [66].

Figure 2.13 verifies our expectation that employed fabrication technologies for implants are lagging behind mainstream ones. We anticipate this lag to be primarily due to three reasons:

1. **Technology availability**: The lack of access of the academic and research community to the most recent fabrication processes at the time of design.

2. **Robust designs**: The need to develop reliable and safe devices targeted for medical use while at the same time limiting implementation costs. Designers have often preferred a stable (at the time) process technology, i.e. a technology node one or more steps behind the then top node. Especially,

these days that *process variation* is becoming a serious problem for transistor devices, reliability is expected to play an even more important role.

3. **Analog-design limitations**: The limited usefulness of small-feature-size fabrication processes in analog design may have been an inhibiting factor for implant designers.

The above reasons are common phenomena in most academic/research environments, yet we have no full-proof way of verifying any of them: Research-group internal policies as well as their particular financial status and design mentalities are involved. However, this *technology delay* appears to be narrowing over time, as implant trends are converging fast towards mainstream ones. The convergence is anticipated to be (partly) due to the following reasons:

1. The previously observed shift from full-custom to COTS-based designs, as shown in Section 2.6.1; effectively, from systems designed from scratch (thus, riskier) to ones designed from pre-verified components.

2. The gradual maturing and streamlining of design tools and processes in the general microelectronics field which has significantly shrank design and development times in all engineering fields, including implantable systems.

Continuing on the issue of implant physical properties, it would also be interesting to investigate the implant physical dimensions; that is chip(set) surface area, packaging volume and weight. Reported data are even sparser in this case, thus conclusions will be drawn very carefully. First off, the *'total chip(set) area'* attribute has been divided in two subsets one including overall chip-area figures and the other overall chipset-area figures. Overall median[9] chip size is $20.75\ mm^2$ and chipset size is $5.38\ cm^2$. Chip-, chipset-size and package-volume trends have been estimated in Research Question 2.5 and are displayed in Figure 2.14. In all three plots, the sample size is too small to draw robust conclusions. Yet, from the first plot we can deduce that, while chip size has shrank considerably over the early years, it has reached a plateau and even somewhat increased over the more recent years.

At first glance, this observation might be unexpected given the miniaturization trends of fabrication technology, as illustrated in Figure 2.13. However, this

[9]To limit the influence of data skewness as well as outliers, median values are generally reported as measures of central tendency.

Question 2.5 *How does implant* chip area*,* chipset area *and* package volume *change over time?*

The relation of each of the scale variables 'total chip area', 'total chipset area' and 'package volume' with time (scale variable 'publication year') is being explored. In a fashion similar to 'fabrication-technology miniaturization', a least-squares, quadratic curve has been fitted for the first dependent variable and is plotted along with 95% CI's in Figure 2.14a. From the scarcity of points in the scatterplot, the low availability of chip-area data (20 cases) is apparent. The sample size of 'total chipset area' is even smaller (14 cases). Analysis indicates the best fit to be the line illustrated in Figure 2.14b. Chipset-area samples are scarce but appear to have smaller dispersion in this case, as can be seen in the same figure.

'Package-volume' data (28 cases) has been fitted with a least-squares, quadratic regression line too, as shown in Figure 2.14c. However, a Log-Likelihood-Ratio (G) test ($G = 3.3464$, $p = 0.1876$) reveals that the fitted quadratic curve (in fact, all 1st-, 2nd- and 3rd-degree curves) is not statistically different from the intercept-only line. This implies that the passage of time has no (detected) appreciable effect on the implant package volume. This result can actually be also visually verified from Figure 2.14c: The green, dashed line representing an intercept-only model (equal to the mean of package sizes) never goes out of the CI band.

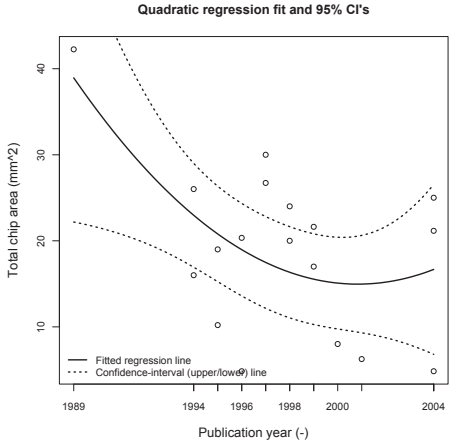
increase in chip size can be explained by the rise of COTS-based implants, as shown in Research Question 2.2. If this is indeed a strong driving factor[10] for the observed increase in chip size, one would not expect it to manifest also on the general implant dimensions, that is, on the chipset size and on the package size. Surely enough, Figure 2.14b reveals a monotonically dropping trend over time. Even through sample size prohibits drawing definite conclusions, the primary technical reason behind this trend is thought to be the ability to integrate increasingly more discrete components (digital and analog) on single chip dies. As a result, chipset real estate is shrinking with every improvement in microelectronics and MEMS technologies.

A lot of previously bulky discrete components such as current drivers, ADCs/-DACs, even hybrid capacitors and coils can these days be fully co-integrated on chip and sensor/actuator structures can be fully micromachined on the same die where the PCC lies, effectively building whole so-called Systems-on-Chip (SoC's). This signifies another driving factor of increasing chip sizes and a relation with decreasing chipset sizes. Unfortunately, we can not explore this quantitatively due the lack of sufficient information (i.e. cases where both figures are known).

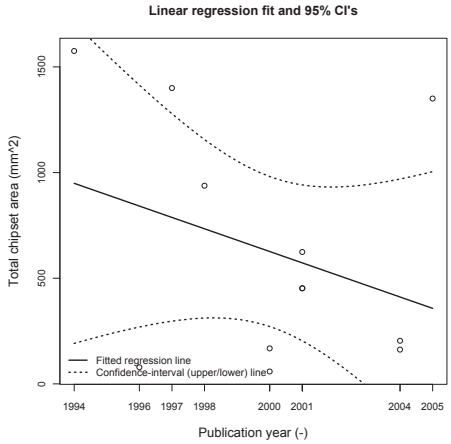
Implant *'package volume'* displays an overall median of $10.20\ cm^3$ and trends over time are also depicted in Figure 2.14c[11]. The volume appears to be shrink-

[10] There are more suspected driving factors for this trend and will be discussed in the following sections, as more survey data is analyzed.

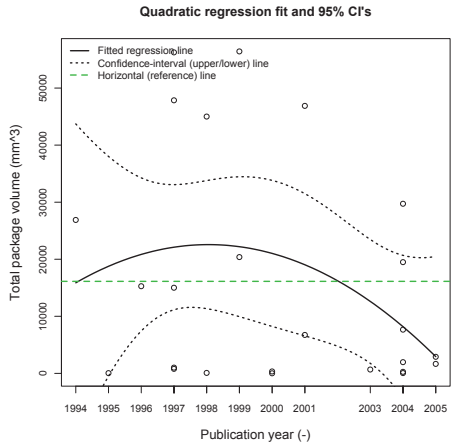
[11] By reflecting on the average implant package size (and the chipset size, above), one can realize that these dimensions are very close.



(a) Chip-area (mm^2).



(b) Chipset-area (mm^2).



(c) Package volume (mm^3).

Figure 2.14: *Implant physical-dimensions trends over the years.*

ing over the years but, as Research Question 2.5 has revealed, the change is not significant. In effect, implant package size has not shrank appreciably over time. Given the fact that these are research-level implantable devices and given the small average size of roughly $10\ cm^3$ they exhibit, even for non-commercial devices, this stagnancy in packaging sizes is understandable.

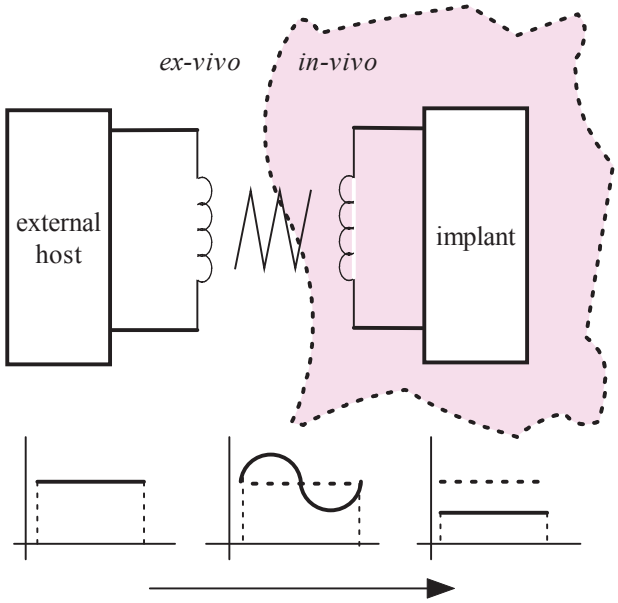


Figure 2.15: *Schematic representation of RF-induction principle.*

2.6.3 Power features

Power consumption is cardinal for the design of implantable systems and has, therefore, been studied separately. The implant 'power source' comes in one of two flavors: it is either an included miniature *battery cell* (55% of all cases) or an *RF inductive link* (45% of all cases) established by an external host transmitter which transfers (induces) electromagnetic power wirelessly, in the form of an RF carrier signal, to the implant (see Figure 2.15). As roughly sketched in the figure, a fraction (typically 10-20%) of this signal is captured and AC/DC-converted (rectified and smoothened) by the implant for generating a stable DC supply.

Rapid improvements in chemical technology have resulted in smaller form-factor, larger-capacity batteries with none of the problems of older systems, like the high-rate self-discharge or the hydrogen-gas emission of Mercury-Zinc power cells. Lithium-Iodine power cells, the latest addition in a long list of Lithium-based batteries, achieve nowadays high chemical stability, very long shelf-life, high energy density and gradual (as opposed to abrupt) depletion [118]. What is more, the absence of gas emissions has seriously eased the task of hermetic sealing of implants.

Compared with induced power, cell-generated power has the extra benefits of providing very stable output and requires no patient intervention. Conversely, RF induction requires some (careful) intervention on the part of the patient. More importantly, it is highly dependent on the material as well as the distance and alignment between the coupled coils, with minor variations in any one of the three seriously affecting the link quality and, thus, the delivered power. The exposure of living tissue to RF waves is an additional consideration for power induction, which places upper safety bounds to the amplitude and frequency of allowable electromagnetic radiation.

For such reasons, most prominent commercial implantable systems, such as pacemakers and ICDs, have exclusively utilized battery cells as their power source. Further, the advances in battery technology in conjunction with a given power budget for these systems has resulted nowadays in, for instance, pacemakers with a lifetime of more than 7 or 8 years.

The limited power rate sustained by RF induction as well as the finite capacity of battery cells makes employing techniques for *low-power* implant operation crucial. The use of low-power commercial components in off-the-shelf-built implants or the careful IC design for reducing power in custom-built ones are the rule in almost all studied cases. Also, the use of low duty-cycle digital signals moving around a device is a common practice for many a researcher (e.g. McCreesh et al. [89, 90]). For this reason, such provisions have not been explicitly reported. Instead, the focus has been placed mainly on *architecture-level* and even *system-level* techniques. Low-power mechanisms encountered in this study appear mainly in the following flavors:

i. interrupt-triggered **power-save modes** (e.g. sleep, standby, off);

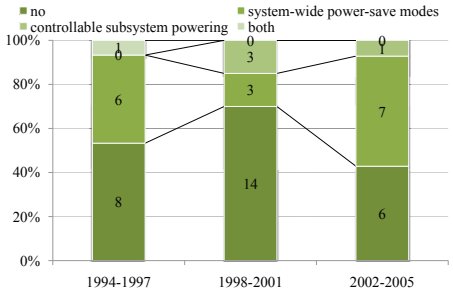
ii. **controllable pulse-powering** of implant subsystems depending on their functionality; and

iii. firmware implementations of **adjustable operational settings** (e.g. sampling rate) when there is no need for maximum performance.

In the crosstabs of Figure 2.16a, low-power provisions have been grouped into four major categories for analysis: (a) system-wide power-save modes, (b) controllable subsystem powering, (c) both provisions, and (d) no provision whatsoever. As expected, the bulk of such techniques has been encountered in battery-powered systems where prolongation of operational lifetime is of primary concern. Namely, out of 29 battery-powered devices (3 unspecified),

	no prov.	power-save modes	subsystem powering	both	TOTAL
battery	9	15	3	2	29
RF induc.	20	4	1	0	25
TOTAL	29	19	4	2	54

(a) *Type of power source used and associated low-power provisions.*



(b) *Low-power-provisions trends over the years.*

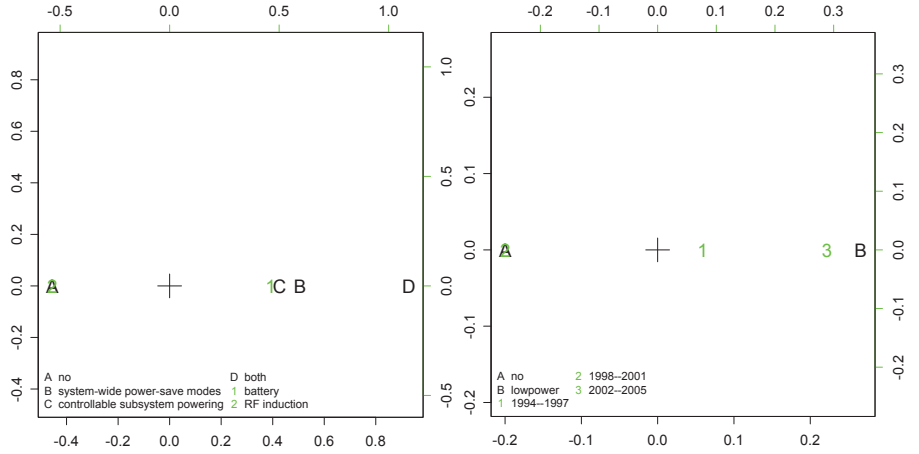
	1994-1997		1998-2001		2002-2005	
	Pi	$\pm\sigma$	Pi	$\pm\sigma$	Pi	$\pm\sigma$
no	0.533	0.129	0.700	0.102	0.429	0.132
system-wide power-save modes	0.400	0.126	0.150	0.080	0.500	0.134
controllable subsystem powering	0.000	0.000	0.150	0.080	0.071	0.069
both	0.067	0.064	0.000	0.000	0.000	0.000
#N	**15**		**20**		**14**	

(c) *Probability estimates (p_i) and their standard errors SE (σ_{p_i}).*

Figure 2.16: *Implant low-power provisions.*

21 present some kind of low-power provisions, while only 5 out of 25 RF-powered devices (3 unspecified) take some similar action. It is interesting to notice also that, generally speaking, the most commonly encountered low-power technique has been *power-save modes* for battery-based and RF-based systems alike.

Figure 2.16b illustrates the percentage of implants equipped with such provisions over the years. We have accounted for the coreless bias manifesting in



(a) *Low-power provisions vs. power source.*

(b) *Low-power provisions vs. publication year.*

Figure 2.17: *CA plots of low-power provisions vs. power source and publication year.*

Question 2.6 *Do implant* low-power provisions *depend on the* power source *employed?*

The strong correlation between implant low-power provisions and employed power source is verified by a chi-square test which is significant at the .01 level (X^2 = 13.3176, p = 0.0034) as well as a CA plot (Figure 2.17a).

the period 1998–2001 as a small rise in devices with no provisions. However, the ratio of implants displaying low-power techniques does not appear to have changed significantly albeit for a slow change in the type of provision employed (from *power-save* to both *power-save* and *subsystem-powering*). This is also verified through Research Question 2.7. Given that battery-powered implants do not appear to be increasing over the years[12], this would cause no reason for concern. However, we will see, next, that overall implant power consumption appears to be increasing over time, which may entail a potential, future hazard for implant design.

Question 2.7 *Does the percentage of implants with* low-power provisions *change over time?*

We have run a chi-square test investigating the trends of implants with low-power provisions over time. The test is not significant (X^2 = 2.6056, p = 0.2718); thus, there is no supporting evidence that the number of implants with low-power techniques has changed over the years.

[12]Refer to the technical report [133] for more details.

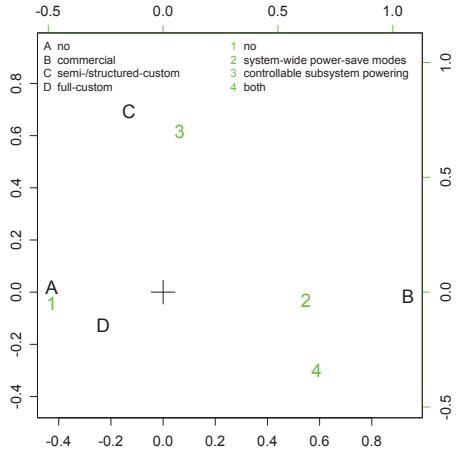
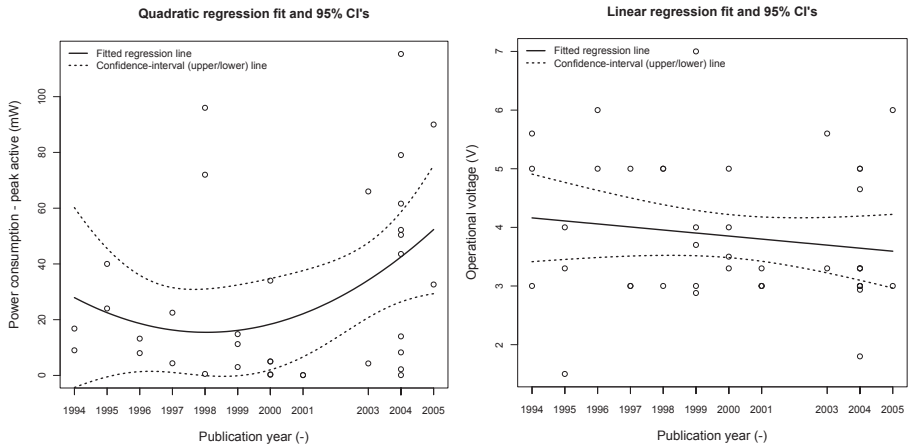


Figure 2.18: *CA plot of implant low-power provisions vs. PCC design.*

Question 2.8 *Does implant* PCC design *have an effect on the* low-power provisions *employed?*

We perform a chi-square test which shows significance at the .05 level ($X^2 = 18.0109$, $p = 0.0351$). To explore this relation further, in Figure 2.18 we have drawn a CA plot. We can see a strong affinity of *subsystem-powering* techniques for semi- or structured-custom PCCs but since this relation is based on a single case, we do not analyze it further. We can also observe a strong relation between *power-save* techniques and commercial PCCs. As will be shown in Section 2.6.5, this is largely due to the increasing use of commercial μCs/μPs which typically come with one or more low-power states built in. Lastly, custom-built PCCs are the ones most obviously lacking any sort of low-power technique. This agrees with common sense: custom designs are usually optimized ones with less needs for explicit power saving.

Having discussed power-source types and low-power techniques for implants, it is now time to see what the actual power consumption of the studied devices is. An attempt has been made to distinguish between the power consumption of the digital and that of the analog part of each presented implant. However, this has not been possible in most of the cases since researchers do not explicitly mention separate figures for those. Another sort of partitioning of the power figures has also been attempted based on the functional state of the device, namely: *standby*, *average* and *(peak active)* power consumption. Even though the sample size for the first two groups has been prohibitively low for extracting robust trends over the years, the last group, peak active power consumption has yielded some interesting results. Along with operating voltage, analysis of power consumption is handled in Research Question 2.9.



(a) *Power consumption – peak active (mW).* ***(b)*** *Operating voltage (V).*

Figure 2.19: *Implant power-consumption and operating-voltage trends over the years.*

Question 2.9 *How does implant* power consumption (peak active) *and* operating voltage *change over time?*

We wish to explore the relation of each of the scale variables 'power consumption (peak active)' and 'operating voltage' with time (variable 'publication year'). A least-squares, quadratic curve has been fitted for the first dependent variable and is plotted along with 95% CI's in Figure 2.19a. While dropping in the middle period of the study, probably due to the *coreless bias*, the curve rises in the latest period – an unexpected trend. For the variable 'operating voltage', a linear regression model has been deemed most suitable and is plotted in Figure 2.19b. This trend is in agreement with shrinking fabrication technologies (see discussion in Section 2.6.2).

The Research Question, above, has produced an unexpected finding: As Figure 2.19a illustrates, peak power consumption is actually increasing over the years. The observed trend is counter-intuitive since implant designs are expected to display shrinking power profiles as technology matures. By closely studying the survey study cases, we anticipate this upward power trend to be the combined result of *two opposing drivers*. The first one, pulling power consumption up, is thought to be caused by the following phenomena:

i. There is a tendency (common in other fields of microelectronics) to add as many features to new implantable systems as possible, rendering them multifunctional devices (to be revisited later). More features means more transistors and, thus, more power. In favor of this argument consent the

previous findings on 'total transistor count' and 'chip area'.

ii. As will be explained in Section 2.6.5, implant designs are increasingly using μC- and μP-based PCCs over FSM-based ones. While μCs and μPs provide multiple benefits such as high flexibility to implantable systems, they also come with a somewhat higher power requirement than ASIC designs, in the general case.

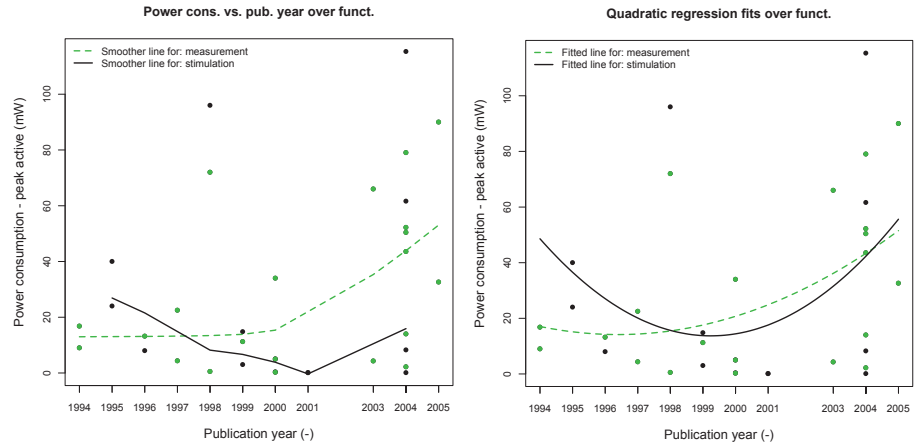
iii. The advances in chemical technology have led to higher-capacity, smaller-volume batteries. As a result, designers have increasingly indulged to the temptation of building newer systems with power needs exceeding those of older ones since new battery technologies allow it.

The other driver, relaxing power consumption, is thought to originate from the following phenomena:

i. The lowering of the operating voltage (following technology miniaturization) has led to reductions in power consumption.

ii. The gradual refinement over the years of low-power techniques in all aspects of implantable devices (PCC, communication module, analog front-ends, interfaces etc.) has contributed heavily in keeping power consumption in check. To exemplify, the monitoring system designed by Wouters et al. [152] consumes about $72\ \mu W$ while monitoring and a mere $13.85\ \mu W$ in standby mode. This constitutes an impressive 5-fold reduction in power consumption.

Except for peak active power consumption, we also attempted to investigate standby and average power figures but the collected data is prohibitively little. Nevertheless, we have calculated cumulative power-consumption figures for each different power state in order to appreciate the impact the previously discussed low-power modes of operation have on power expenditure. Peak active power has a median value of 13.600, average power of 0.325 and standby power of 0.045. It can be concluded that standby power is about an order of magnitude lower than average and as much as three orders of magnitude lower than peak power. Obviously, average power consumption in a device depends largely on its duty cycle (i.e. standby-active ratio) which, in turn, depends heavily on the application at hand.



(a) *Smoother lines.* ***(b)*** *Fitted quadratic lines.*

Figure 2.20: *Power-consumption trends over the years for different implant functionalities.*

Question 2.10 *How does* power consumption (peak active) *change over time for different implant functionalities?*

At this point in our analysis it would be interesting to investigate whether implants from the two major functionalities (measurement and stimulation) exhibit different power-consumption trends over time. The question can be addressed through use of a *dummy-variable regression model* (as discussed in Section 2.5.4.4). 'functionality' has been used as the dummy (independent) variable for this model and the scale independent variable 'publication year'.

We revisit Figure 2.19a and highlight with different colors the two categories of the dummy variable. A new scatterplot is, thus, created; see Figure 2.20a. On the scatterplot, smoother lines have been drawn to assist exploration. Quadratic lines have been fitted separately on stimulation and measurement points and the result is illustrated in Figure 2.20b. The figure reveals that – contrary to stimulation implants – measurement implants do not have a noticeable dip in power consumption in the middle period 1998–2001. This supports our initial claim that the coreless bias is mostly caused by a certain number of ocular stimulation implants in the particular time period.

Visual inspection reveals that both implant types exhibit an eventual increase in their power needs over the years. However, the rise of the two fitted curves is not identical and we wish to test whether the difference between them is statistically significant. Therefore, we perform a Log-Likelihood ratio test which yields non-significance ($G = 1.0080$, $p = 0.7993$). This outcome means that the two implant functionalities do not differ significantly in their power-consumption trends over time. This, in turn, implies that we have correctly selected the unified regression line in Figure 2.19a to cumulatively describe the implant power trends.

Question 2.11 *Does the chosen implant* functionality*,* PCC design*,* PCC type *and* power source *have an impact on its* power consumption (peak active)*?*

Having acquired detailed power figures from our analysis, it is also interesting to see whether power consumption is generally affected by implant functionality (generally speaking, not over time), PCC design as well as PCC type. Based on the discussion of Section 2.5.4.5, exploration has been performed visually, through boxplots; see Figure 2.21.

Figure 2.21a reveals that stimulation implants exhibit slightly higher power profiles when active. However, the difference is insignificant, as has been verified by a Kruskal-Wallis (KW) rank sums test ($X^2 = 0.7964$, $p = 0.3722$). This means that – in the absence of opposing data – measurement and stimulation implants could be treated singularly in terms of their power requirements at design time.

As far as PCC design is concerned, the boxplots in Figure 2.21b follow common sense: the further we depart from full-custom design towards COTS-based PCCs, the higher the power consumption (at least, the peak active) is. Of course, implants with no PCC at all consume the lowest power overall. PCCs originating from commercial components not only have the higher power budget but also exhibit the wider dispersion of power profiles, making implant design with such components not only power costly but also difficult to predict. Even though these differences are easily visible in the figure, a KW test returns non-significance ($X^2 = 3.9675$, $p = 0.2650$).

The last comparison to make here for power consumption is between the two possible power sources for implants: batteries or RF induction (Figure 2.21c). It can be observed that battery-powered devices consume, on average, more than RF-based ones. Based on a KW test, this difference does also not appear to be statistically significant ($X^2 = 1.7432$, $p = 0.1867$). Given the fact that all four boxplot-based observations tend to agree with common sense, we are inclined to consider the large survey-data "noise" as the main culprit for the non-significant test outcomes.

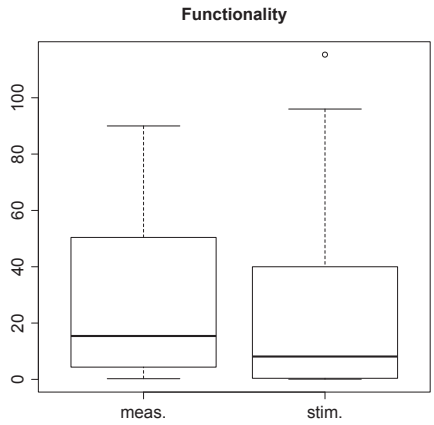
2.6.4 General implant features

Information about subsystems commonly met in all implantable devices is presented in this section. The first studied attribute is the I/O peripherals, that is, any biosensors, bioactuators, stimulating/measurement electrodes or other interface to the living tissue. Information on the number and specifics of the various peripherals has not been always available, therefore, some assumptions have been made for allowing statistical analysis[13].

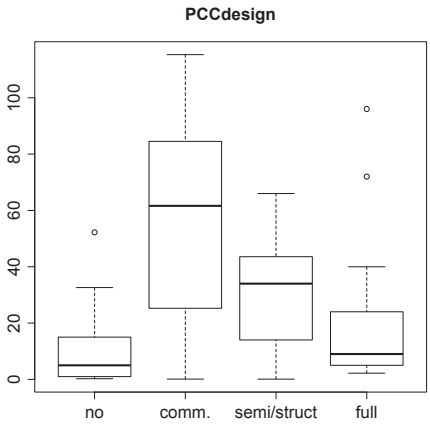
Question 2.12 *How does the number of* peripherals per implant *change over time?*

We wish to fit proper regression curves on the number of implant peripherals over the years but we wish to do so separately for both primary categories (measurement, stimulation) of the 'functionality' variable. LS quadratic curves have been fitted for each category and results have been plotted in Figure 2.22. Stimulation peripherals per implant exhibit a monotonously dropping trend over time. On the other hand, measurement peripherals appear to roughly remain unchanged in number. This has been verified by a G test ($G = 0.7693$, $p = 0.6807$) and visually illustrated through an intercept-only (green dashed) line in Figure 2.22b.

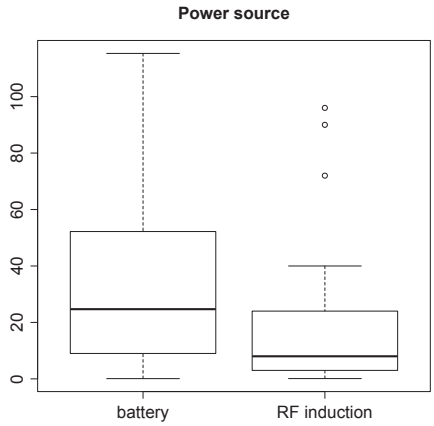
[13] See Technical Report [133] for more details on the subject.



(a) *Power cons. vs. Functionality*



(b) *Power cons. vs. PCC design*

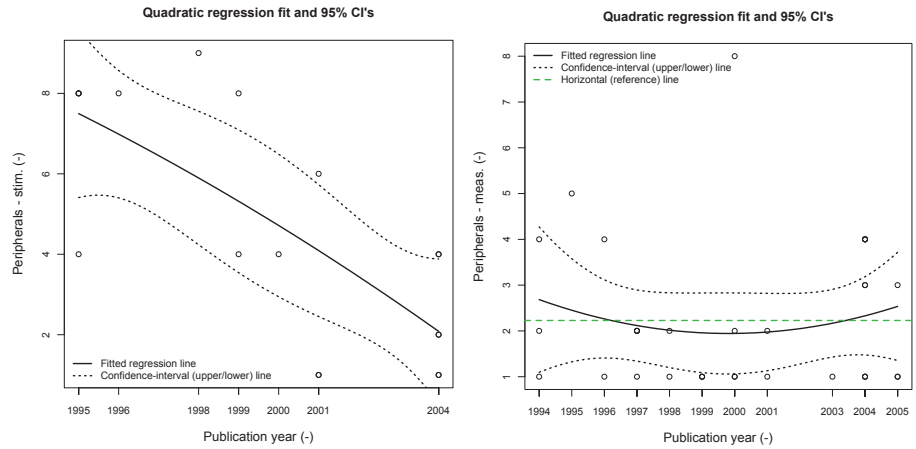


(c) *Power cons. vs. Power source*

Figure 2.21: *Boxplots of implant power consumption (peak active) with respect to different device characteristics.*

As Research Question 2.12 reveals, implants have maintained a more or less constant number of measurement peripherals over the surveyed time period. On the other hand, the number of stimulation peripherals is slowly decreasing. The reasons behind this drop are not clear, though they could be attributed to the observations that more recent implants tend to focus on stimulating isolated nerves or muscle bundles. In so doing, perhaps they can be placed in "tighter" locations inside the body and stimulate at a finer granularity and accuracy.

There is another potential reason behind these trends: More recent implants



(a) *Stimulation-peripheral count.* ***(b)*** *Measurement-peripheral count.*

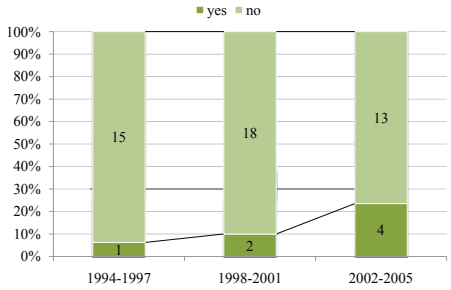
Figure 2.22: *Number of peripherals per implant over the years.*

opt to include both sensory and actuating peripherals in them to achieve applications with autonomous, closed-loop control. This approach imposes more stringent design constraints, and since the average number of measurement peripherals was already low to begin with – 2 to 3 per implant –, the resource that could be limited is the number of stimulation peripherals – from about 8 down to 2 per implant.

Besides, if we also take into account the previous analysis revealing rising power-consumption trends, implant designers may have found themselves (consciously or not) struggling not only for *smaller implant sizes* and *more complex processing* but also for a *tighter power budget*. Reducing the number of actuating elements on the implant may have been a (partial) solution to the latter problem, as well. The phenomenon on combined peripheral types in recent implants is investigated in Research Question 2.13, below.

Question 2.13 *Do the relative percentages of implants with* both sensory and actuating peripherals *change significantly over time?*

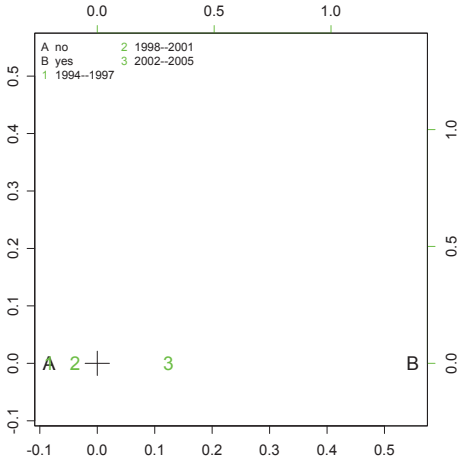
A chi-square test is run between the categorical variables 'peripheralsboth' and 'publication year' (grouped). The test does not reveal a strong correlation ($X^2 = 2.4352$, $p = 0.2959$) between the variables, which is understandable given the estimated sample standard errors (Figure 2.23b). However, Figure 2.23a shows a slow but sure rising trend for implants combining both types of peripherals. Therefore, we have also plotted a CA (Figure 2.23c) plot which visually confirms that such implants are more related to the latest survey period 2001–2005 than the earlier periods.



(a) *Trends over the years.*

	1994-1997		**1998-2001**		**2002-2005**	
	Pi	$\pm\sigma$	Pi	$\pm\sigma$	Pi	$\pm\sigma$
yes	0,063	0,061	0,100	0,067	0,235	0,103
no	0,938	0,061	0,900	0,067	0,765	0,103
#N		16		20		17

(b) *Probability estimates (p_i) and their standard errors SE (σ_{p_i}).*



(c) *CA plot of number of implants with both types of peripherals vs. publication year.*

Figure 2.23: *Implants including both actuating and sensory elements over the years.*

It is telling of this shift towards implants with more sophisticated control also that many monitoring or stimulating implants support *independent* sampling or driving – respectively – of their peripherals as well as calibration, compensation and power control. Along the same lines, our analysis of the data shows that a large number of implantable systems is built with some degree of *modularity* for being able to accommodate a number of different peripherals over the same sensing or actuating channels. Thus, to a limited extent, reusability of designs has been pursued by various researchers like, for instance, Lerch et al. [83] and Valdastri et al. [140]. *Adjustability*, *modularity* and other advanced implant features will be revisited in detail in Section 2.6.6.

We, next, move to study the internal-processing capabilities of implants. The attribute 'internal processing' refers to those implants that actually perform some kind of *non-standard* processing in their PCC (if present). As initially defined, under the term "processing" we do not include signal-manipulation tasks commonly encountered in implants such as data sampling, filtering, A/D-conversion, multiplexing and the like. Rather, we wish to isolate any extra processing tasks an implant core might bear like, for instance, the 2-D motion detection algorithm based on readouts from two accelerometers, implemented by Wouters et al. [152].

The survey reveals *data-related* and *control-related* tasks. Data-related tasks mainly comprise sensor-data manipulation, namely *data compression*, *reduction* and *encoding* as well as implementation of various *algorithms* such as the motion-detection scheme, mentioned above. Also, *data-integrity operations* such as CRC-checksum calculation and parity checks. The latter category entails in-system, on- or off-line control such as *self-diagnostics*, *power-supply monitoring* and *closed-loop control* of peripherals. This observation seems to support our previous analysis on reducing implant peripherals. Such an instance is the case of a drug-delivery implant developed by Cross et al. [25] to adjust the oestrus cycle of cows. Another, popular instance of closed-loop control actively performed by PCCs is the rate-responsive stimulation of the cardiac or various skeletal muscles, i.e. stimulation dynamically adapted to the heart or respiration rate, respectively. Heart-rate-responsive pacemakers have been present in the market for more than two decades now. A third case of encountered closed-loop control is the dynamic adjustment of the sensor sampling rate, based on the monitored levels of biological quantities (e.g. significant fluctuations in blood pressure trigger an increase in sampling rate).

Statistics show that, in an overall, only 20% (12 out of 59 devices, one unknown) of all studied systems features some kind of non-standard processing.

The obvious reason for this poor percentage is the extra power expenditure that extra processing entails. So, even if an application could make good use of some sort of data manipulation inside the implant, designers have generally preferred to telemeter the data to an external host system and have it process the data. At times, they have even transmitted results back to the implant for closed-loop control, as in the case of Smith et al. [127].

However, studying the number of processing-enabled implants over the years reveals an interesting trend (see Figure 2.24). Implants equipped with internal-processing capabilities display a slow albeit steady increase. The figure also indicates control-related tasks to be increasing faster than data-related ones.

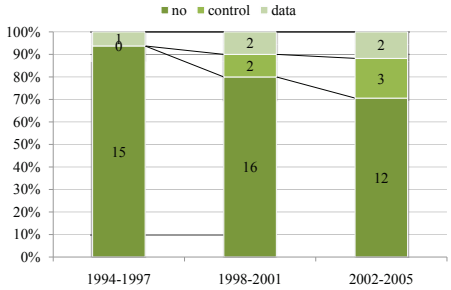
Question 2.14 *Do the relative percentages of implants supporting* internal-processing tasks *change significantly over time?*

Over the total number of studied systems, the ratio of implants performing processing tasks in-vivo is small, as can also be seen from Figure 2.24a. Statistical analysis reveals practically no correlation (X^2 = 3.5585, p = 0.4690) between this ratio and the passage of time (see also CA plot in Figure 2.24c). Perhaps the change is not significant enough to register, yet the fact remains that from a single processing-enabled implant in the period 1994–1997, the number has increased to 5 in the period 2001–2005.

As far as implant memory circuitry is concerned, results from the 'internal data-storage capability' variable reveal that 27 out of 59 devices (46%) feature some kind of memory module either as a discrete chip or as part of the PCC chip. Of these 27 devices, 15 are measurement while 12 are stimulation implants (see crosstabs in Figure 2.25a).

Question 2.15 *Does the chosen implant* functionality *have an impact on its* internal-data-memory size*?*

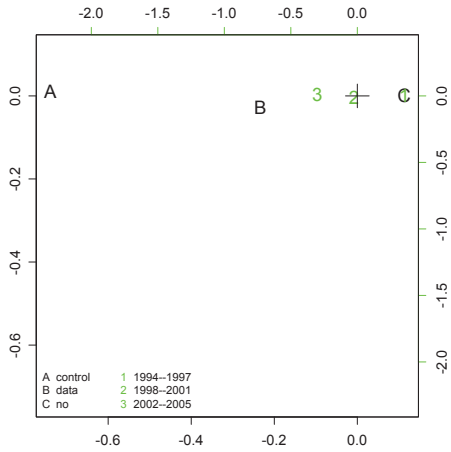
The boxplots in Figure 2.25b reveal a twofold difference between the – otherwise small – data-memory sizes of measurement (312 *Bytes*) and stimulation (176 *Bytes*) implants. They also reveal a larger dispersion of memory sizes for measurement implants. We do not expect the difference in size to be statistically significant and this is supported by a Kruskal-Wallis rank sums test (X^2 = 0.0722, p = 0.7881) which strongly rejects the null hypothesis. Therefore, in the absence of further data, implant design should not be differentiated in its memory requirements based solely on its functional purpose. There is no opposing data to the fact that both measurement and stimulation implants require similar amounts of memory, albeit for different purposes.



(a) *Internal-processing trends over the years.*

	1994-1997		1998-2001		2002-2005	
	Pi	$\pm\sigma$	Pi	$\pm\sigma$	Pi	$\pm\sigma$
no	0.938	0.061	0.800	0.089	0.706	0.111
control	0.000	0.000	0.100	0.067	0.176	0.092
data	0.063	0.061	0.100	0.067	0.118	0.078
#N	**16**		**20**		**17**	

(b) *Probability estimates (p_i) and their standard errors SE (σ_{p_i}).*

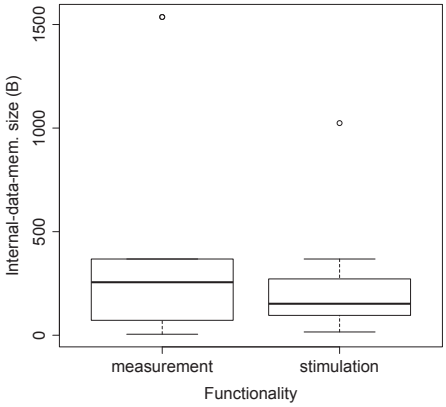


(c) *CA plot of number of implants supporting internal-processing tasks vs. publication year.*

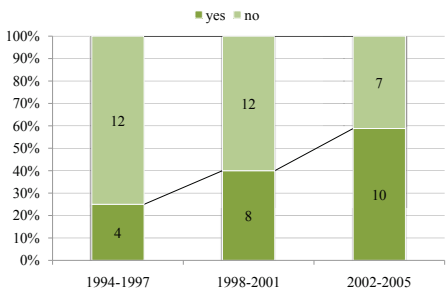
Figure 2.24: *Internal-processing-enabled implants over the years.*

	internal mem. present	no internal mem. present	TOTAL
meas.	19	15	34
stim.	13	12	25
TOTAL	32	27	59

(a) *Type of implants incorporating in-system data-memory blocks.*



(b) *Boxplots of data-memory sizes for stimulation and measurement implants.*

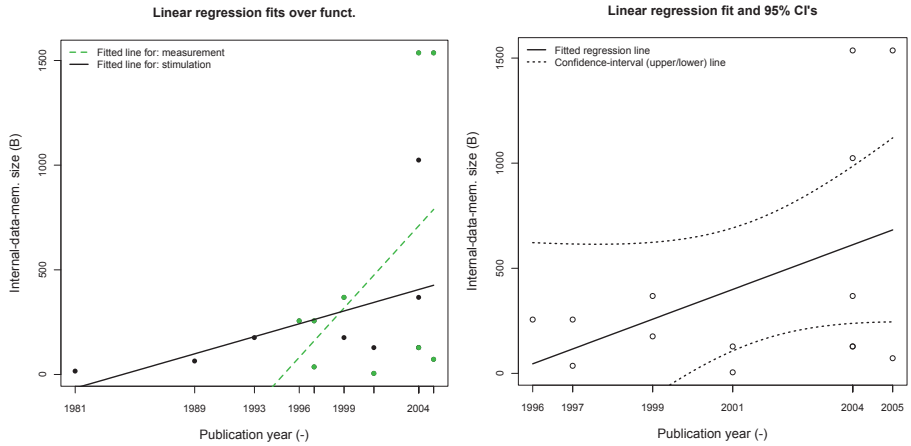


(c) *Data-memory-availability trends over the years.*

	1994-1997		**1998-2001**		**2002-2005**	
	Pi	$\pm\sigma$	Pi	$\pm\sigma$	Pi	$\pm\sigma$
yes	0.250	0.108	0.400	0.110	0.588	0.119
no	0.750	0.108	0.600	0.110	0.412	0.119
#N	**16**		**20**		**17**	

(d) *Probability estimates (p_i) and their standard errors SE (σ_{p_i}).*

Figure 2.25: *Implant internal-data-memory availability and size.*



(a) *Fitted lines per functionality.* ***(b)*** *Fitted aggregate line.*

Figure 2.26: *Internal-data-memory trends over the years.*

Question 2.16 *How does* internal-data-memory size *change over time for different implant functionalities?*

Research Question 2.15 has shown that implant data-memory sizes are, in an overall, similar for both measurement and stimulation implants. However, we would further like to investigate how these memory sizes change for each functionality category over time. Once more, a *dummy-variable regression model* has been fitted to the data. Linear models have been fitted on both stimulation and measurement data and are illustrated in Figure 2.26a. We can immediately make two observations: (a) both implant categories show increasing data-memory sizes with time, and (b) measurement-implant memory sizes exhibit a bigger slope than stimulation ones, which is expected given the formers' strong data-collecting nature.
Besides, we wish to test whether the difference between them is statistically significant. Keeping in mind that data is, as usual, scarce, we perform a G test which yields non-significance ($G = 1.8384$, $p = 0.6066$). Therefore, a regression line (Figure 2.26b) for both implant categories should suffice to describe data-memory trends over the years.

Although data-memory sizes are increasing over time, they are markedly low compared to the mainstream-market size, even after accounting for the highly constrained nature of implantable systems. The reluctance of researchers to pack more data memory with their designs may be attributed to the rising power and chip-area issues. Still, the latest achievements in high-density, low-power memories [66] can lead to diminished penalties and, thus, implant implementations with larger capacities in the future. This expectation is backed by survey findings which indicate a steady increase of memory-enabled devices every 4 years (see Figure 2.25c).

An increase in storage capacity shall allow implanted devices to become actual in-vivo data-loggers with the ability to store and even process large quantities of gathered biological data. It shall also untie the hands of ADC- and DAC-unit designers, allowing them to implement higher-resolution converters that run little risk of overflowing the larger implant memories. In short, memory-capacity increase is anticipated to favor not only the autonomy but also the processing capabilities of future implantable devices. This has already been observed in commercial pacemakers which incorporate ever larger memories to accommodate for increased functionality [118].

Let us now discuss the supported sampling rates of implantable systems. Most biologically-generated, electrical signals display low frequencies (up to $1\ kHz$), e.g. ECG and EEG signals range from $0.05\ Hz$ to $100\ Hz$. Thus, implants with sampling frequencies of $2\ kHz$ (according to the Nyquist theorem) are sufficient. This is the case in this survey, too, as captured in the attribute 'sampling rate'. There are few exceptions in the range of tens of kHz and even fewer ones in the range of hundreds of kHz. Most of these are maximal settings of specific systems rather than nominal settings.

High sampling frequencies are also justified by the designers' wish to extract not only peaks in a physiological signal but also further, detailed values within the spectrum, for high-fidelity visualization purposes. This is particularly the case nowadays with EEG neural signals like, for instance, the neural-activity recording implant proposed by Mohseni and Najafi [96] which supports sampling rates of up to $8\ kHz$. As previously mentioned, such high sampling frequencies can be adjustable either at design time or at run time. In the latter case, fast sampling intervals are triggered when activity above a specific threshold is detected in the monitored biological quantity(-ies).

Question 2.17 *How do supported measurement-implant* sampling rates *change over time?*

Based on analysis of the collected data, a LS quadratic curve has been fitted (see Figure 2.27). Supported sampling frequencies appear to grow over time, however, they seem to be reaching a plateau over the more recent years. The figure shows some exceptionally high frequencies ($250-300\ kHz$) achieved recently, yet the bulk of devices lies at lower frequencies ($< 10\ kHz$). Still, these frequencies should be sufficiently high for most biomedical applications which might explain the approaching plateau.

Lastly, the discussion in this section focuses on ADC and DAC units, both commonly found inside implantable devices. In case an implant contains a PCC, these components are usually built or chosen with a resolution equal to the data-word size of the PCC, since the ADC output (or DAC input) is typically fed to (or supplied by) the PCC. The designers' wish to acquire (or

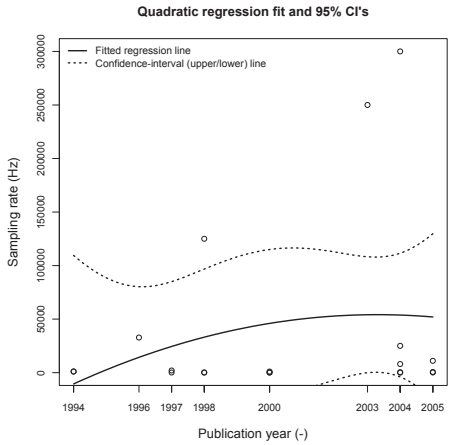


Figure 2.27: *Implant sampling-frequency trends over the years.*

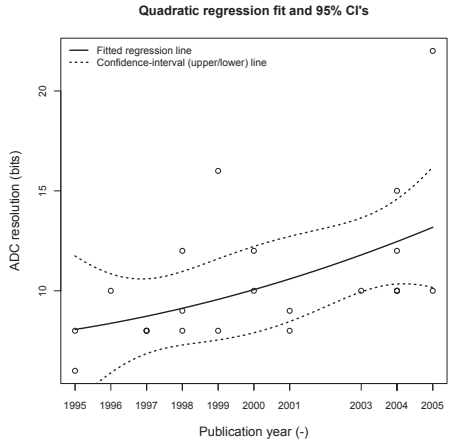
generate) as high-resolution signals as possible is limited by the processing power and memory size they can fit in a single implant.

Reported figures on ADC/DAC resolution are very erratic with ADCs in the range from 6 to 22 *bits* and DACs in the range from 3 to 8 *bits*. Respective median values are 10 *bits* and 5 *bits*. This large dispersion of values once more underlines the largely customized design of many implants and their PCCs which support various precision ranges depending on the medical application at hand.

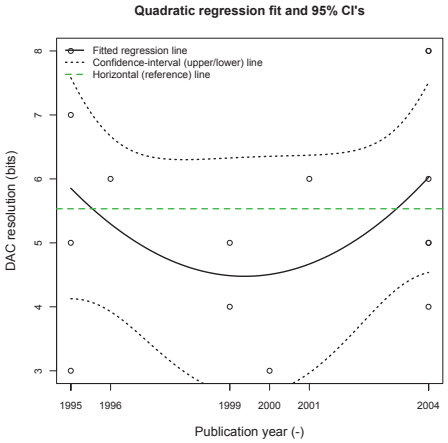
Question 2.18 *How does* ADC and DAC resolution *change over time?*

Quadratic curves have been fitted to the ADC- and DAC-resolution data points, over time. Figure 2.28a depicts a monotonously increasing trend in the number of implemented ADC bits. On the contrary, Figure 2.28b depicts a trend which – after conducting a G test – turns out to be not significantly different from the intercept-only line ($G = 2.4117$, $p = 0.2994$); that is, DAC resolution exhibits no notable change over the years of the study.

Having discussed ADC/DAC-resolution trends, it would be interesting to associate them with the trends we have discovered so far on the number of peripherals per implant (cf. Figure 2.26a) and on the data-memory sizes (cf. Figure 2.22b and Figure 2.22a). We have combined these trends in a single table (Table 2.28c) where we have used the following illustrative notation: '↗' denotes an upward trend in time, '↘' denotes a downward trend in time and '0' denotes no appreciable change in time. According to the table, it appears that current measurement implants have reached a sufficient range of monitored physiological quantities and the designers' primary goal these days is to pro-



(a) ADC resolution.



(b) DAC resolution.

implant type	peripheral count	data-mem size	ADC/DAC resolution
measurement	0	↗ ↗	↗
stimulation	↘ ↘	↘	0

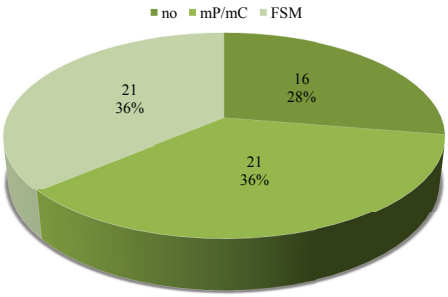
(c) Qualitative comparison of the implant-peripherals' trends over time.

Figure 2.28: *ADC and DAC resolution trends over the years.*

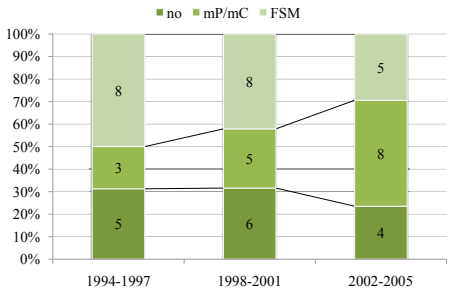
vide higher-quality readouts. Such readouts will require considerable amounts of memory to handle and store. Stimulation implants appear to follow a different trend. Stimulation-signal resolution is high enough these days but the effort is placed on achieving more focused and fine-tuned stimulation (perhaps of isolated nerves or muscle bundles). A concurrent increase in memory sizes would help increase device autonomy and available stimulation patterns.

2.6.5 PCC features

Since the implant PCCs are of special interest in this study, the current section is concerned with the PCC specifics. The 'PCC architecture' attribute queries implants with respect to the style and complexity of the PCC architecture used; that is, μController, μProcessor or FSM. As discussed in Section 2.6.2, 42 out of 58 (2 unknown) (70%) devices in this study feature some type of PCC. The



(a) *Overall PCC-architecture distribution.*



(b) *PCC-architecture trends over the years.*

	1994-1997		1998-2001		2002-2005	
	Pi	$\pm\sigma$	Pi	$\pm\sigma$	Pi	$\pm\sigma$
no	0.313	0.116	0.316	0.107	0.235	0.103
mP/mC	0.188	0.098	0.263	0.101	0.471	0.121
FSM	0.500	0.125	0.421	0.113	0.294	0.111
#N	**16**		**19**		**17**	

(c) *Probability estimates (p_i) and their standard errors SE (σ_{p_i}).*

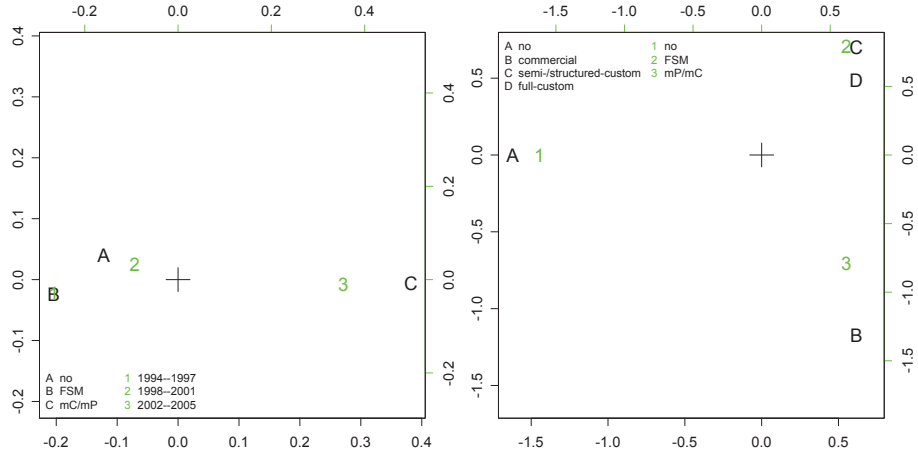
Figure 2.29: *Implant PCC architecture.*

remaining 30% contain no PCC of any sort and their functionality is hardwired. Some of these cases include implants which need to be extremely miniature in size and are designed for a very specific task, e.g. ICP measurement with the implant placed in the space between the cranium and the brain.

Going back to the PCC-enabled cases, half of them (35%) utilize FSMs as cores. This fact alone further propagates our thesis that implants are highly dedicated designs. The remaining half consists of custom-designed or commercially available μC/μP-based PCCs[14]. Overall distribution of PCC architectures is depicted in Figure 2.29a.

Another observation is revealed through Figure 2.29b, illustrating the *relative* percentage of encountered PCC architectures per time period of the study. The bar chart reveals an increasing number for μC/μP-based implants at the cost of a receding number of FSM-based ones. Although Research Question 2.19

[14] Although not exactly the same, in this analysis, we have grouped μC and μP cases in a single category due to the small number of μP cases in the survey.



(a) *Publication year.* ***(b)*** *PCC design.*

Figure 2.30: *CA plots of PCC architecture vs. PCC design and publication year.*

Question 2.19 *Do the relative percentages of implant* PCC-architecture *categories change significantly over time?*

We perform a chi-square test to verify the change of PCC-architecture trends over the years. The test returns a non-significant p-value (X^2 = 3.4746, p = 0.4817) suggesting that the observed shift in Figure 2.29b may be coincidental. Based on our analysis to this point, we do not think this is the case. Unfortunately, the standard error is too high to draw solid conclusions. Still, to further explore the evolution of PCC-architecture trends, we have also generated a CA plot of the two variables (Figure 2.30a). The plot does indeed support, if visually, our observations: μC/μP-based solutions are more akin to the latest time period 2001–2005 while FSM-based ones to the earlier time periods.

has not proven a statistically significant shift in implant PCC architectures over the years, our previous findings (e.g. increasing peak power consumption) and some upcoming ones (e.g. high correlation between certain PCC architectures and low-power provisions (Research Question 2.22)) hint otherwise.

In view of the above, the primary reason for the observed trend in Figure 2.29b is anticipated to be the following: As many researchers clearly state in their published work, their preference towards μP or μCs stems from their desire to build flexible, adaptable systems which allow online – that is, *in-vivo* – adjustment or fine-tuning of their operation. Reasons for doing so typically are post-operative, patient-specific implant adaptations, re-calibration of drifting sensory/actuating peripherals after chronic implantation and so on. Leveraging

such implant flexibility is considered important enough that drawbacks like increased device size and reduced testability are gladly traded over it [117]. The flexibility issue will be revisited in more detail in Section 2.6.6.

By combining the last observation with the trends on PCC designs observed in Figure 2.10b, it is further suspected that these μC/μP-based PCCs are mostly commercial, off-the-shelf ICs (by Microchip, Atmel, Motorola, Philips etc.). The following Research Question 2.20 verifies this claim.

Question 2.20 *Does the implemented implant* PCC architecture *have an effect on the* PCC design *selected?*

To answer these questions, a chi-square test is run between the 'PCC architecture' variable and 'PCC design'. The test returns a high X^2 and significance at the .001 level (X^2 = 84.1921, p = 4.853e − 16). To investigate the exact nature of the correlation, a CA plot has been drawn in Figure 2.30b. As the plot reveals, all used FSMs are semi-custom- or structured-custom – and, to a smaller extent, full-custom – implementations. Employed μC/μPs, on the other hand, are mostly commercial components.

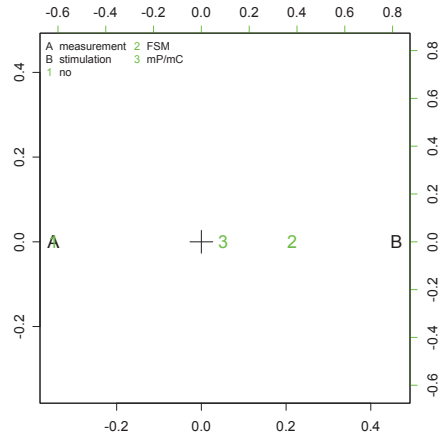
The above relation between the PCC architecture and the PCC design could be explained by the fact that recent microelectronics-technology advances in transistor sizes and performance have gradually allowed for "coarser" than custom (ASIC) approaches to be viable in implant design. This correlation may, in fact, be signaling a general shift in the implant design paradigm altogether. Implant designers appear nowadays more and more willing to trade some extra implant area, performance and/or power for introducing (a) higher device properties, (b) simplifications in the design cycle, and (c) increased safety – all provided through use of commercial ICs with established, mature design cycles and preverified, pretested cores.

Besides, by carefully employing low-power techniques such as low-power states and selective powering of subsystems (as discussed in Section 2.6.3), μC/μP-based PCCs can narrow the power gap from the more economic FSM-based PCCs further. This has been rigorously exemplified in the work by Salmons et al. [117] whereby two functionally equivalent instances of a single stimulation implant have been built and compared, one as an FSM-based and the other as a μC-based device.

We, next, wish to explore whether a connection exists between the implant functionality and the PCC architecture implementing it. In effect, Table 2.31a has been generated. From the table it can be seen that μC/μP-based implants have been employed equally for both kinds of functionalities, which underlines their suitability for the whole range of (surveyed) implant applications. Conversely, implants incorporating no PCC have been almost exclu-

	meas.	stim.	TOTAL
μC/μP	10	10	20
FSM	8	13	21
no PCC(s)	14	2	16
TOTAL	32	25	57

(a) *Crosstabs.*



(b) *CA plot.*

Figure 2.31: *Implant PCC architecture vs. functionality*

sively employed for measurement purposes. This is justified mainly due to the large number of minimalistic implementations designed to simply monitor and telemeter one or two physiological parameters such as temperature and glucose levels. More complex functionality than this is seldom implemented in pure hardware.

On the other hand, FSM-based devices have been slightly more often preferred for stimulation purposes, as the table reveals. There are two main reasons suspected for this. First, their reduced size – compared to μC/μP-type PCCs – is more suitable for the class of applications where the microstimulator needs to occupy as little space as possible, e.g. inside the ocular cavity, interwoven in the body of a muscle and so on. In short, custom design (shown to primarily be used for FSMs) is favored in cases where size is the highest concern. The second reason is related to the electrical properties of the system at hand. FSM-based stimulators can be built with special process technologies to *inherently* support specific electrical properties e.g. high-voltage tolerance or large-current generation, like BiCMOS. In measurement systems where sensors typically operate at small voltages and currents, such special processes are unnecessary. However, that does not exclude μC/μP-based systems from being realized in special process technologies.

To explore the relation between different PCC-architecture types and power consumption, Figure 2.32 has been put together. In Table 2.32a is shown the preference of designers towards a battery-based or an RF-induction-based

Question 2.21 *Does implant* functionality *have an effect on the* PCC architecture *implemented?*

To back the discussion above and to also complement Research Question 2.3, we wish to investigate implant 'functionality' with respect to 'PCC architecture'. We run a chi-square test which shows significance at the .05 level (X^2 = 11.4179, p = 0.0223). A CA plot (Figure 2.31b) confirms the test and our observations that stimulation implants mostly contain FSM-based cores while measurement implants are either coreless designs (which agrees with our earlier findings on PCC types) or contain μC/μP-based cores.

power source depending on the type of PCC architecture they are using for their design. Given that batteries can usually provide more energy (per time unit) than RF-induction methods, it is clearly seen that they are the common choice for the more power-hungry μC/μP-based PCCs. For the more conservative FSM-based PCCs, RF induction is a highly viable option resulting in smaller-sized devices.

Question 2.22 *Does implant* PCC architecture *have an effect on the* power source *and* low-power provisions *employed?*

We wish to find out whether the relation between the PCC architecture and the power source as well as the low-power provisions chosen is significant. A relation between FSMs and RF-induction as well as μCs/μPs and batteries has been established in Table 2.32a and is also manifested in Figure 2.32b. Yet, the chi-square test returns a marginally non-significant p-value (X^2 = 5.6411, p = 0.0596).

Conversely, when low-power provisions are concerned (see reported schemes in Table 2.16a), the test is strongly significant at the .001 level (X^2 = 32.1188, p = 1.548e − 05). The associated CA plot in Figure 2.32c reveals that FSMs are most often encountered implementing either subsystem-powering schemes or no schemes at all, whereas μCs/μPs usually implement only power-save or both reported schemes. It should be noted that this affinity to the power-save scheme stems chiefly from the fact that most utilized μCs/μPs are commercial components with various low-power operational modes already built-in.

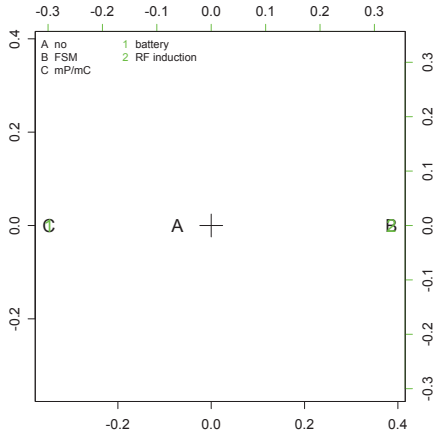
While on the power-consumption topic, we also explore the effect the various PCC-architecture types have on the reported power consumption (peak active) of the surveyed implants. To illustrate, we have drawn Figure 2.32d. The boxplots reveal a visible difference in median values across coreless, FSM-based and μCs/μP-based implants.

Question 2.23 *Does the chosen implant* PCC architecture *have an impact on its* power consumption (peak active)*?*

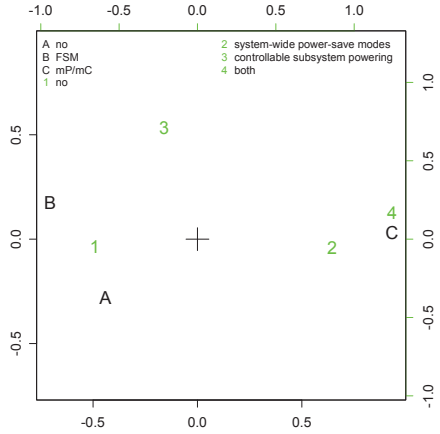
Although the various PCC types rank as expected in their power requirements (i.e. μC/μP > FSM > no PCC), the difference in medians is, once more, not supported by statistics – a Kruskal-Wallis test returns a non-significant p-value (X^2 = 3.2052, p = 0.2014).

	battery	RF ind.	TOTAL
μC/μP	14	6	20
FSM	7	14	21
no PCC(s)	9	7	16
TOTAL	30	27	57

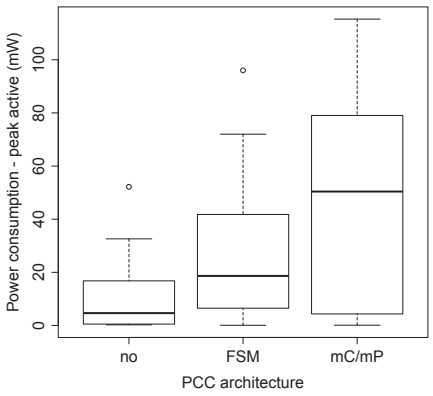
(a) *Implant PCC architecture vs. power source.*



(b) *CA plot of PCC architecture vs. power source.*



(c) *CA plot of PCC architecture vs. low-power provisions.*



(d) *Boxplot of implant power consumption with respect to different PCC architectures.*

Figure 2.32: *Implant PCC architecture vs. power features.*

Basic clock frequencies for the PCCs (attribute 'PCC frequency'), commonly range from 1 to 10 *MHz* with an overall median of 4 *MHz*. If a PCC is not present in a design (e.g. the design is analog in nature), then overall-system frequencies have been collected where available ('non-PCC timing'). They generally appear to be one order of magnitude lower than their digital counterparts with an overall median of 600 *kHz*. This is somewhat expected for analog

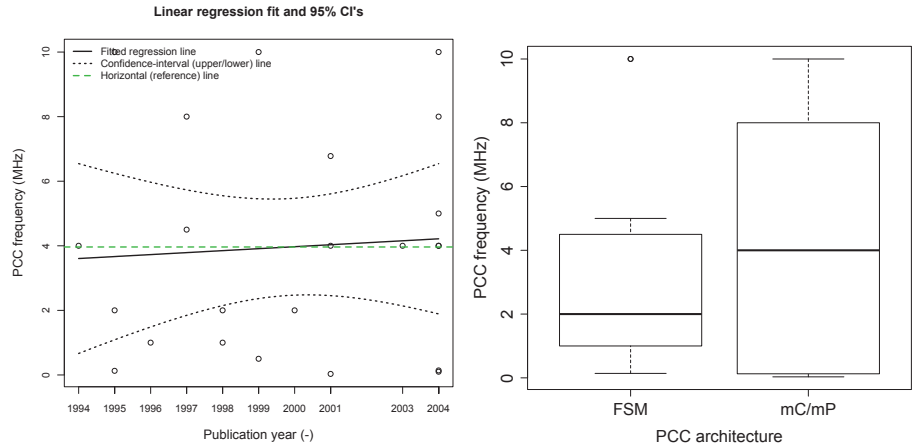
or mixed-signal designs where timing is used only for driving sampling-related sub-systems such as switched-capacitors, analog MUXes and simple counters. Analog circuits are, after all, more susceptible to clock skews as frequencies grow. PCC-enabled (thus, mainly digital) designs are - on the other hand - more tolerant to noise and voltage drifts and allow easily higher clock rates. There is, though, an additional, more subtle reason anticipated for this gap. Interestingly, designers are "forced" to go to higher clock rates due to the above discussed tendency towards commercial ICs which typically run at a minimum of a few MHz, e.g. the Atmel AVR datasheets indicate a $4-MHz$ μC clock.

Question 2.24 *How does implant* PCC frequency *change with different* PCC architectures *and over time?*

Except for median values, it would also be interesting to investigate whether PCC clock frequencies exhibit any appreciable shift in values over the survey studied years. Please note that the sample size of non-PCC-timing data is not large enough to allow analysis.
In Figure 2.33a we have fitted a LS line through the PCC-frequency data points. The standard error of the available data is high, thus, reducing the certainty of our conclusions. With this in mind, the observation can be made that PCC frequencies are – in fact – not picking up over the years. This has been verified by a G test which shows no significant difference ($G = 0.0951$, $p = 0.9536$) from an intercept-only line. It could be explained by the fact that operating frequencies are already high enough for present implant applications to be served. Irrespectively – or, perhaps, because – of this, designers may have chosen to hold operating frequencies more or less stable in an effort to cut down on consumed power. In either case, frequencies do not appear to be changing appreciably with time which implies that provided processing throughputs (and sampling rates) are sufficient for current applications. This conclusion is also supported by Research Question 2.17 on stabilizing sampling frequencies over the years.
In order to appreciate also the range of operating frequencies across different PCC-architecture types, we have plotted Figure 2.33b. The boxplots reveal a higher median value but also a wider dispersion of values for μCs/μP-based PCCs compared with FSM-based ones. The difference in medians (approx. $2\ MHz$) does not appear to be significant ($X^2 = 0.1471$, $p = 0.7014$).

PCC instruction-set statistics have also been collected and presented in this section; namely, the actually *used* and the *total* number of instructions as well as the instruction-word and data-word sizes. Reported instruction sets vary greatly and may consist of 1 up to 130 different instructions, with the median around 25.5. Outstanding cases with 100 or more instructions are commercial PCCs of which the implant utilizes a rather small subset for performing its functions. However, in a few cases, designers have also reported the actual subset of total instructions available that they have used in their design. This subset has a median value of 2.5 instructions. In effect, an overall ratio of used to total instructions can be calculated, roughly equal to 11%. In Research Question 2.25, instruction-count trends over the years are also investigated.

The low instruction-utilization ratio (overall and across the years) underlines



(a) *PCC-frequency trends over the years.*

(b) *PCC-frequency vs. PCC architecture.*

Figure 2.33: *PCC-operating-frequency characteristics with respect to PCC architecture and trends over the years.*

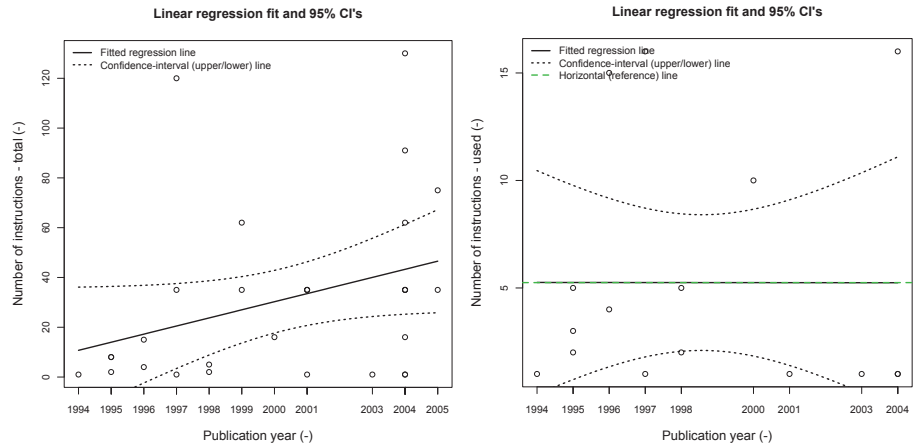
Question 2.25 *How does the number of total and used* instructions *of implant PCCs change over time?*

Except for overall median values, we have attempted to visualize how *total* and actually *used* instruction-set sizes have evolved over the surveyed years. In Figure 2.34, linear regression models have been fitted on the available data. The reader can observe that sample sizes for used instructions over the two latest time periods are very small; thus, we should be very careful when extracting conclusions.

Trends of the total number of instructions appear to be climbing over the years (Figure 2.34a), reflecting the introduction as PCCs of commercial, μCs/μPs PCCs featuring increasing instruction-set sizes. Trends of the number of used instructions, on the other hand, appear to be static (Figure 2.34b). We have performed a G test which returns an insignificant p-value of almost one (X^2 = 1.0632e − 05, p = 0.9999). There is, therefore, no evidence that the number of used instructions is changing over the years. By combining information from the two trend lines – i.e. a linearly increasing number of total instructions and a stagnating number of used instructions –, we can conclude that the instruction utilization ratio is dropping linearly over the studied period.

the price paid for employing commercial PCCs instead of custom-built ones. It, further, suggests that introducing (off-the-shelf) implant PCCs with smaller instruction sets could help to maintain, in the future, all the benefits of commercial ICs while limiting unnecessary hardware complexity and, thus, power, area and delay.

The last attributes discussed in this section are instruction-word and data-word



(a) *Total PCC instructions over the years.*

(b) *Used PCC instructions over the years.*

Figure 2.34: *Number of used and total instructions per implant PCC over the years.*

sizes. For the former, a median count of 14 *bits* and for the latter of 8 *bits* has been found. Instruction words are relatively erratic in their sizes ranging from as few as 4 *bits* to as many as 45 *bits*, being heavily dependent on the application at hand as well as on the communication protocol implemented between the implant and the external host. This is because usually an instruction word is a subset of one (or more) command frame(s) transmitted to the implant. Data words are in most cases equal to 8 *bits*, which is nothing more than the adherence to the "standardized", popular memory-word sizes of 8 *bits*. This is becoming more apparent by the fact that most implants make heavy use of the on-chip data memory of their commercial PCCs (being mostly $8-bit$ architectures so far).

2.6.6 Miscellaneous implant features

Having reviewed general implant characteristics and the specifics of PCCs, a discussion on more "qualitative" attributes of implantable systems is in order. These attributes – adjustability, versatility, programmability, modularity and reliability – are more difficult to detect in the description of the various designs, let alone quantify their effects. This makes statistical analysis (so far limited by low sample populations and large noise margins) even more difficult to

perform. Even so, the present taxonomy and analysis would be incomplete were such attributes to be disconsidered.

The first trait, termed 'adjustability', is related to the ability of an implant to support diverse operational settings for its included peripherals. That is, monitoring devices can poll their attached sensor(s) at different sampling rates for data acquisition, stimulating devices can generate different current-amplitude ranges for their stimulation channels and so on. The benefits of building adjustable systems are many:

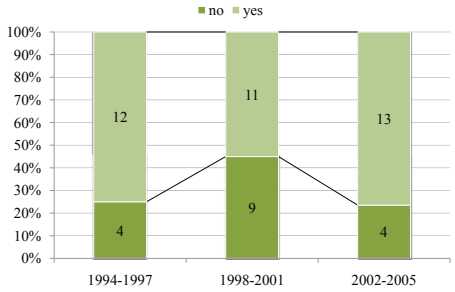
1. Various instances of the same system can be produced with (slightly) different operational settings and at no extra development costs.

2. System operational settings can be altered *after* implantation for carefully adapting the device functions to the special needs and comfort of each patient, e.g. the stimulating patterns of a microstimulator can be fine-tuned, depending on the responses by the specific patient.

3. In many a case, physicians do not know *a-priori* what the "correct" settings to a medical problem are and, thus, need to first implant the device and then adjust its functionality by trial and error. Furthermore, dynamic, in-system adjustment of functional parameters gives in-vivo medical studies a great boost by allowing researchers and physicians to easily test various parameter combinations (e.g. stimulation trains) in test subjects without the need for repeated implantation and explantation surgeries which are tedious, expensive, unpleasant and often dangerous ordeals.

4. Further, adjustability of peripherals allows for newer, more sophisticated versions of implemented algorithms – which make use of these peripherals – to be seamlessly accommodated in a system. The result is enhanced functionality, e.g. in the case of ICDs, for sensing and classifying arrhythmias [147].

5. Lastly, adjustable systems can compensate for readout-circuitry nonlinearities (due to transistor mismatches etc.), sensor drifts, biological-tissue reactions and other phenomena which manifest only after device fabrication and packaging or, worse, after chronic implantation inside the body. It has been shown that digital compensation can improve sensor accuracy by at least an order of magnitude over the uncompensated device. Compensation can be achieved algorithmically, in the PCC of an implant, e.g. polynomial-coefficient generation can be set up and run automatically with negligible delays during a testing phase [149].



(a) *Trends over the years.*

	1994-1997		1998-2001		2002-2005	
	Pi	$\pm\sigma$	Pi	$\pm\sigma$	Pi	$\pm\sigma$
no	0.250	0.108	0.450	0.111	0.235	0.103
yes	0.750	0.108	0.550	0.111	0.765	0.103
#N	**16**		**20**		**17**	

(b) *Probability estimates (p_i) and their standard errors SE (σ_{p_i}).*

Figure 2.35: *Relative distribution of implants featuring adjustable peripherals over the years.*

In effect, adjustable systems feature increased useful lifetimes, higher flexibility and sharper operation. It is stressed that most of the above benefits hold for systems that provide dynamic, i.e. in-system, in-vivo tweaking of their parameters rather than static, i.e. assembly- or packaging-time adjustments.

Analysis of the survey data reveals that researchers have indeed identified early on the need to build adjustable devices and have, thus, incorporated various features to support it. Overall, 42 out of the 60 surveyed implants have been found to offer some kind of adjustability. This makes up for as high as 70% of all cases and is indicative of the unfailing attention researchers have placed on building adjustable systems. In many cases, these are second- or third-generation systems, augmented with dynamic settings, instead of preset ones.

Figure 2.35a shows adjustability-enabled implant trends over the years. The small drop in adjustable systems observed over the period 1998–2001 may be attributed to the previously discussed coreless bias. Generally speaking, we see a stable inclusion of features for adjustability over the years.

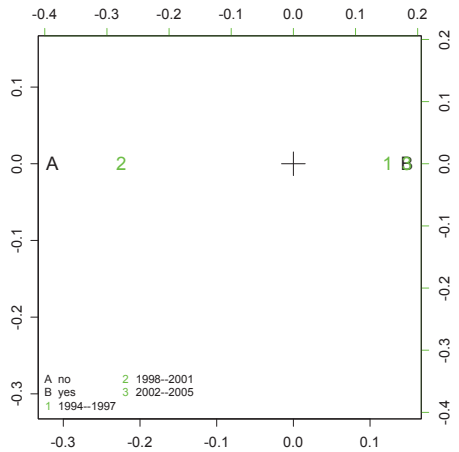


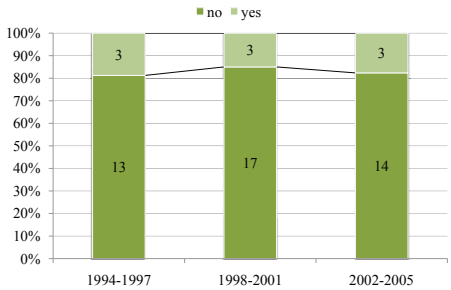
Figure 2.36: *CA plot of implant adjustability vs. publication year.*

Question 2.26 *Do the relative percentages of* adjustable *implants change significantly over time?*

Given the more or less constant presence of implants with adjustability features over the years, we do not expect any noticeable change in the already high percentage of such systems over the years. Indeed, we perform a chi-square test which gives no evidence of a changing trend ($X^2 = 2.4709$, $p = 0.2907$). A CA plot of the same data is equally inconclusive, as seen in Figure 2.36.

A small percentage of the studied implants (17%) goes a bit further and features designs that accept a limited gamut of interchangeable peripherals; we have termed this property 'versatility'. More specifically, these implants feature interfaces (channels) with adjustable characteristics, to which a number of different sensor (or other) modules can be plugged in (at design, fabrication or packaging time). They can, therefore, configure their readout electronics to the sensitivity, the speed and, less commonly, the resolution of the ported sensors. Of course, limitations exist in this approach since the 'versatility' feature depends on various aspects such as the physical interface, the data encoding and the powering scheme of a plugged-in peripheral. For example, if a sensor outputs PWM-modulated data but the implant "understands" only PCM signals, direct interfacing between the two is not possible unless extra glue-circuitry is included. It is stressed here that the versatility attribute refers to implants designed to support interchangeable (pluggable) peripherals and not to support, simultaneously, a number of statically connected peripherals, as was discussed in Section 2.6.4.

The gathered data shows that in the vast majority of cases, the interchangeable



(a) *Trends over the years.*

	1994-1997		**1998-2001**		**2002-2005**	
	Pi	$\pm\sigma$	Pi	$\pm\sigma$	Pi	$\pm\sigma$
no	0.813	0.098	0.850	0.080	0.824	0.092
yes	0.188	0.098	0.150	0.080	0.176	0.092
#N	**16**		**20**		**17**	

(b) *Probability estimates (p_i) and their standard errors SE (σ_{p_i}).*

Figure 2.37: *Relative distribution of implants featuring versatile (interchangeable) peripherals over the years.*

peripherals are sensory units since they are the most easy to accommodate. Actuators usually come with more diverse control and powering requirements, making it more difficult to cover different types of them. An interesting case has been presented by Fernald et al. [41]: the implant is designed with a serial-bus architecture able to interface to an arbitrarily large number of different peripherals (sensors, actuators, memory blocks, transceiver units etc.), all interconnected in a daisy-chain fashion.

Oddly enough, the "versatile" implant cases are not a proper subset of the "adjustable" ones. Indeed, as the study indicates, there are three instances whereby different peripherals are supported but their functionality does not appear to be adjustable. The overall percentage of versatile implants is rather low (17%) and this is also manifested in the percentage of implants with such provisions over the years; see Figure 2.37a. This is to be expected for one main reason: Designing for various peripherals increases the complexity of a system and introduces a certain amount of overhead due to the support circuitry required. What is more, there are – as of yet – no widespread industry standards on sensor/actuator interfaces which can guarantee to researchers portability of

their implant designs through a comprehensive, established pool of standard-conforming peripherals to routinely select from.

However, this is not to say that work on standardizing sensor (and actuator) interfaces for other technological niches has not been attempted in the past. Industry-related bus standards focusing on reliable, high-performance communication have been described and commercialized in the past ranging, for instance, from the popular analog $4 - 20\ mA$ current-loop interface to the Profibus-DP, the CANbus (+, 2.0B etc.), the busWorldFIP Fieldbus and the less exotic Ethernet IEEE802.3. Interesting comparisons between various such interfaces can be found, for instance, in [98] and [44]. In the field of low-power, wireless sensor networks (WSNs), agreeing on a specific bus interface has been the topic of long discussions lately, as well. A prominent case is the work done by Najafi, Wise et al. [87,99,149] who have come up with various flavors of standard bus interfaces. A full-duplex, parallel bus known as the Michigan Parallel Standard (MPS) is implemented with 16 lines whereas a more sparing, half-duplex serial bus, the Michigan Serial Standard (MSS) is implemented with just 4 lines. Another featured "intramodule sensor bus" for environmental monitoring comes with 9 signal lines [155–157] and is based on the MSS. It is also very similar to the transducer-independent interface (TII), based on the recently developed IEEE1451 standard for sensor systems [34,81,151]. Zhang et al. [159,160] have come up with a general-purpose, low-noise, low-power bus standard which is hardware-compatible with the TII standard but expands on it supporting multi-node systems and plug-n-play capabilities.

While not all attributes of standardized interfaces such as the above can be exploited for implantable systems, various items such as high reliability and high accuracy can intuitively migrate to biomedical implants. In this context, the versatility percentage, discussed previously, is rather encouraging and is anticipated to increase in the years to come. Besides, the exact figure may in fact be somewhat higher since for some study cases it was rather difficult to discern whether the system was designed to support more peripherals or not.

Question 2.27 *Do the relative percentages of* versatile *implants change significantly over time?*

Contrary to adjustability, the available data does not support that versatility has not been strongly pursued over the years. Percentages remain low throughout the surveyed years of the study. This is verified by a chi-square test which returns a strongly non-significant p-value ($X^2 = 0.0965, p = 0.9529$). A CA plot (Figure 2.38) does not offer more information.

The 'programmability' field includes all those devices that realize their functionality by executing source code stored in a program memory, i.e. (part of) their control is software-based rather than hardwired or hard-coded. This is

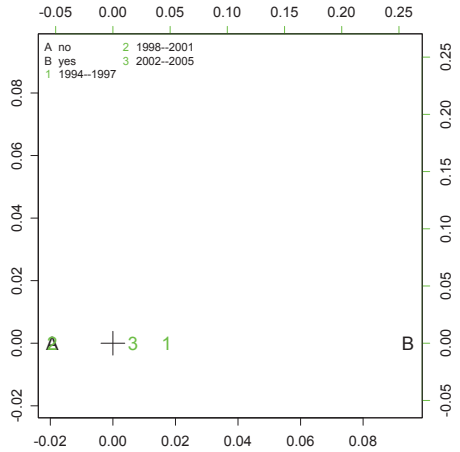


Figure 2.38: *CA plot of implant versatility vs. publication year.*

an interesting-to-know feature since it reveals implantable systems that can potentially adapt their functionality to more than one application by (off-line) reprogramming their control and processing behavior (and by plugging in the proper peripherals).

Data analysis reveals 15% of all cases (14 unspecified) to be programmable, which is exactly the same subset of those implants based on μC/μP PCCs (see Figure 2.29a). This translates to the conclusion that in cases where researchers have wanted to design an implant with programmable functionality, they have almost always opted for a (commercial) μC or μP as the implant PCC.

A prominent instance of a programmable system adapted to two completely different applications is presented in literature by Lanmüller et al. [77]. The first system is designed for cardiomyoplasty, aortomyoplasty, skeletal-muscle ventricle (SMV) and other cardiac-assistance purposes and features ECG-triggered, multichannel stimulation of maximally two skeletal muscles with the goal of increasing muscle output. The second system is a nerve microstimulator developed for use in electrophrenic respiration (EPR) - applied in diaphragm pacing - and for graciloplasty - applied for fecal continence - and allows for activation of maximally two muscles, too. In this case, the ECG-measurement hardware is not required and has, thus, been omitted while the software running in the μC of the implant (as well as the software in the external host computer) has been modified accordingly. Other than that, the system setup remains principally the same. A Motorola μC has served as the PCC in both applications.

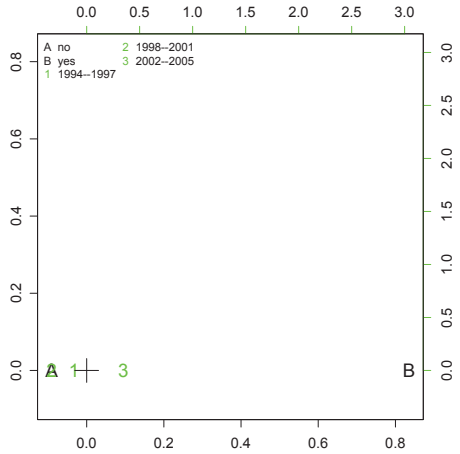


Figure 2.39: *CA plot of implant programmability vs. publication year.*

Over-the-years trends of implants with programmability capabilities are plotted in Figure 2.40a. The trends are not clear but, assuming the dip in the middle period 1998–2001 is due to the coreless bias, programmability-enabled implants appear to be increasing slowly.

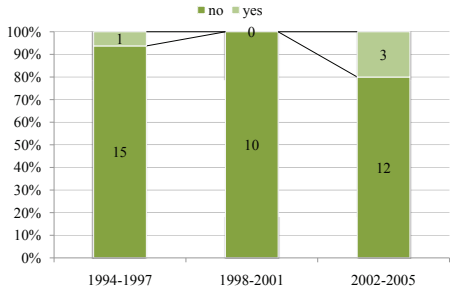
Question 2.28 *Do the relative percentages of* programmability*-enabled implants change significantly over time?*

We wish to investigate whether there is any appreciable change in the number of programmability-enabled implants over the years. A chi-square test reveals no significance ($X^2 = 3.0923$, $p = 0.2131$) which indeed indicates a slow (or totally absent) increase in such devices. A CA plot provides similar visual cues (see Figure 2.39) showing a weak relation between programmability-enabled implants and the latest study period 2002–2005.

Of the programmability-enabled cases, program-memory sizes range from $512\ B$ to $256\ KB$ with a median of $7.3\ KB$. Besides, to investigate program-memory sizes with respect to different implant functionalities, boxplots for stimulation and measurement implants have been plotted in Figure 2.40c.

Question 2.29 *Does the implemented implant* functionality *have an impact on its* program-memory size*?*

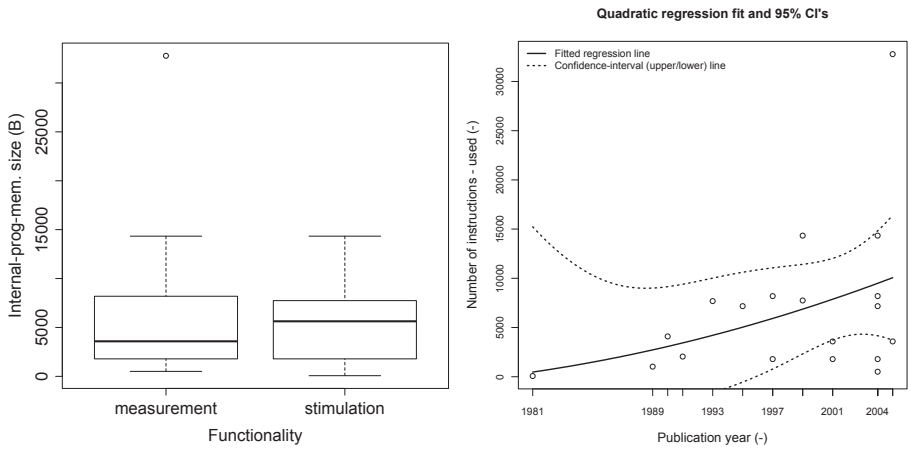
The boxplots reveal that, as was also the case for data memories (see Figure 2.25b), measurement implants display a wider range of values for program memories than stimulation implants, nonetheless with a slightly smaller median value. This underlines the data-intensive operation of the former as opposed to the control-intensive behavior of the latter systems. A Kruskal-Wallis test verifies that the difference in central tendencies is not significant by returning a large p-value ($X^2 = 0.0067$, $p = 0.9347$). This is an interesting finding: existing data does not suggest that either type of implant has significantly larger needs in program-memory size.



(a) *Trends of programmability-enabled implants over the years.*

	1994-1997		**1998-2001**		**2002-2005**	
	Pi	$\pm\sigma$	Pi	$\pm\sigma$	Pi	$\pm\sigma$
no	0.938	0.061	1.000	0.000	0.800	0.103
yes	0.063	0.061	0.000	0.000	0.200	0.103
#N	**16**		**10**		**15**	

(b) *Probability estimates (p_i) and their standard errors SE (σ_{p_i}).*



(c) *Boxplot of program-memory sizes for stimulation and measurement implants.*

(d) *Program-memory size trends over the years.*

Figure 2.40: *Implant programmability capabilities and trends over the years.*

Question 2.30 *How does implant* program-memory size *change over time?*

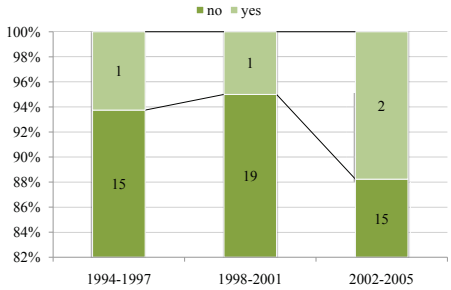
A LS quadratic regression line has been considered the best fit for the available program-memory-size data points; see Figure 2.40d. The regression model reveals a genuine increase of program-memory sizes over the years.

It is interesting to observe that these trends agree with the data-memory-size trends found in Figure 2.26a. From the two figures we can conclude that (a) memory sizes appear to be generally increasing over the years and (b) program memories feature large sizes than data memories. Both these conclusions agree with mainstream memory trends, which supports the current analysis.

Another interesting property of implants is design 'modularity', i.e. it characterizes devices that have been designed generically enough so that their core can be reused in very diverse applications. By the term 'modularity' we don't simply imply those devices that can support a specific set of different peripherals – this was covered by the previously seen 'versatility'. Instead, we consider implants which are based on an architecture capable of supporting a (theoretically) infinite number of application setups, i.e. a general-purpose core which can be used in various biomedical scenarios with no modifications whatsoever.

Only one of the studied implants has been consciously based on a generic (as opposed to highly dedicated) design for the PCC part, introduced by Fernald et al. [41]. As previously mentioned, the implant is designed with a flexible architecture, communication protocol and bus interface for closure over a wide range of peripherals and corresponding applications. The system proposed by Smith et al. [127], which has been also discussed in Section 2.6.1, though not truly generic, comes closer than most other systems with provisions for a large range of measurement and stimulation applications.

If we relax our initial definition of 'modularity' somewhat by including also systems that have been built with sub-blocks intended to be reusable in other scenarios, we come up with 5 more devices (bringing the total to 10% of all studied cases). In these systems, specific circuit modules have been designed to be suitable for a range of different applications. A typical instance is the microstimulator put together by Arabi and Sawan [5] which comprises a voltage regulator, a data/clock separator, a FEC-circuitry and stimulation channels all designed to be reused for implementing other neuromuscular prostheses. Modularity comes also in the flavor of external data/address busses and CS (Chip-Select) lines that allow for expanding a basic system with more, peripheral-to-the-core subsystems if the application demands it. Such a case is presented by Stotts et al. [131]. Over-the-years modularity trends are plotted in Figure 2.41a which indeed illustrates the altogether small number of modular devices.



(a) *Trends over the years.*

	1994-1997		1998-2001		2002-2005	
	Pi	$\pm\sigma$	Pi	$\pm\sigma$	Pi	$\pm\sigma$
no	0.938	0.061	0.950	0.049	0.882	0.078
yes	0.063	0.061	0.050	0.049	0.118	0.078
#N	**16**		**20**		**17**	

(b) *Probability estimates (p_i) and their standard errors SE (σ_{p_i}).*

Figure 2.41: *Relative distribution of implants based on a modular design over the years.*

Question 2.31 *Do the relative percentages of* modularity*-enabled implants change significantly over time?*

We, again, perform a chi-square which returns a non-significant p-value ($X^2 = 0.6579$, $p = 0.7197$). This is expected given the scarcity of the available modular devices. A CA plot (Figure 2.42), however, reveals a similar picture to the programmability attribute before: modular implants are more prominent, if weakly, to the latest study period 2002–2005.

While the design approach sanctioning IP reusability has been tried and proven in other, non-biomedical fields, the low percentage of modular implants revealed by our study indicates that, until now, it has not been truly adopted in microelectronic-implant design. However, we consider this to be an excellent design recipe and well-suited for this field, too. It can potentially offer a tremendous boost to implantable systems by allowing researchers to develop, exchange and utilize IP cores which are guaranteed to work and are already proven in some other implant design. Such reusable and modular designs will reduce effort for clinical approval, will shorten development times and will minimize testing and verification costs, which are non-trivial altogether especially in the biomedical domain where (inter)national regulations are most stringent. Few generic implant designs have been proposed over the last 30

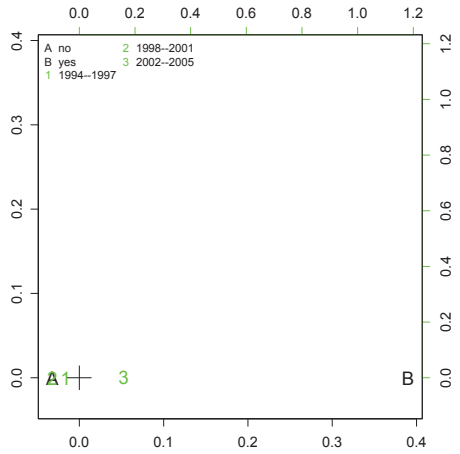


Figure 2.42: *CA plot of implant modularity vs. publication year.*

years or so and, while their significance is not larger now than it was before, recent advances in CMOS microtechnology make generic design more plausible now and allow for new approaches to the subject, previously turned away as unrealistic.

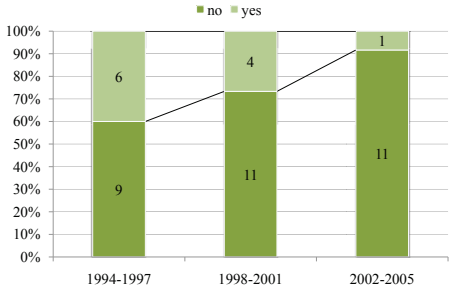
The last parameter studied in this section is 'reliability'. Under this term are included all provisions and techniques employed for increasing the safety and dependability of implantable devices. Design-, implementation-, fabrication- and run-time techniques are considered, intentionally excluding mechanical schemes and focusing on the electrical ones. The reliability parameter is a crucial feature of implantable systems – perhaps the most crucial – given the delicate nature of the biomedical field. Survey findings have revealed various flavors of it which have been grouped in 7 distinct *classes*, as follows:

1. **Duplication** of circuits and structures (multiple threads of supply wiring, backup circuitry etc.);

2. **Self-test/diagnostic circuitry**: battery voltage/temperature, output current/voltage, SW failure, system clock, RF-signal strength, HW breakpoints and debugging etc. implemented through the use of Watch-Dog Timers (WDT) etc.;

3. **Safety circuitry**: unintentional-stimulation prevention, reset on error etc.;

4. **Test/interrogation modes** (SW-based or HW-based, autonomous or externally-controlled);



(a) *Trends over the years.*

	1994-1997		1998-2001		2002-2005	
	Pi	$\pm\sigma$	Pi	$\pm\sigma$	Pi	$\pm\sigma$
no	0.600	0.126	0.733	0.114	0.917	0.080
yes	0.400	0.126	0.267	0.114	0.083	0.080
#N	**15**		**15**		**12**	

(b) *Probability estimates (p_i) and their standard errors SE (σ_{p_i}).*

Figure 2.43: *Relative distribution of implants with reliability provisions over the years.*

5. **Error-detecting/correcting instruction decode**: parity-check, Hamming codes, CRC etc.;
6. Design for **structural testability** (scan-based testing); and
7. **Humidity detection** in hermetically sealed implant packaging.

For such a mission-critical field as biomedical implants are, the overall percentage of devices incorporating reliability-enhancing schemes is surprisingly low, only 35% (17 out of 49 cases, 11 unspecified). Even more surprisingly, trends over the years in fact reveal a dropping ratio of implant devices explicitly designed with some kind of reliability provisions; see Figure 2.43a.

For gaining further perspective on the reliability issue, figures 2.45a and 2.45b have been generated, associating reliability provisions with the PCC type and PCC architecture of implants, respectively. Figure 2.45a indicates that systems with full-custom and commercial PCCs are the most probable to appear with reliability provisions. In both cases, roughly half of the designs are “reliable”. By cross-correlating the figure results with data from the ‘reliability’ variable,

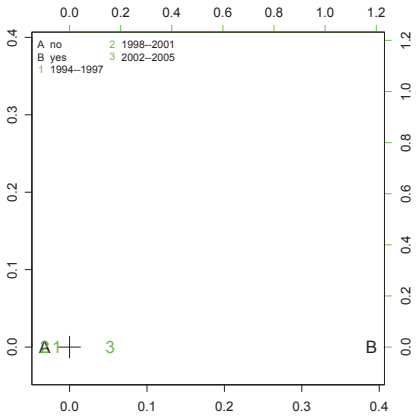


Figure 2.44: *CA plot of implant reliability vs. publication year.*

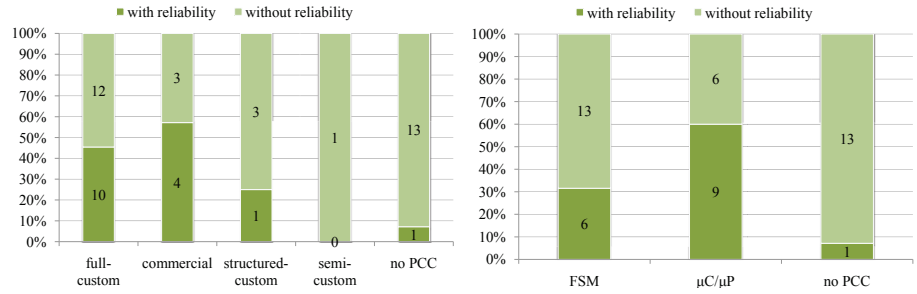
Question 2.32 *Do the relative percentages of* reliability*-enabled implants change significantly over time?*

Statistical analysis, once more, does not verify (X^2 = 3.461, p = 0.1772) the visual observations made from Figure 2.43a. Although scoring a lower p-value than for the previous attributes, p is still not significant. Still, a CA plot (Figure 2.44) tells the same story as the bar chart; i.e. implants with "no" reliability provisions are to be found in the locus of the more recent years of the study.

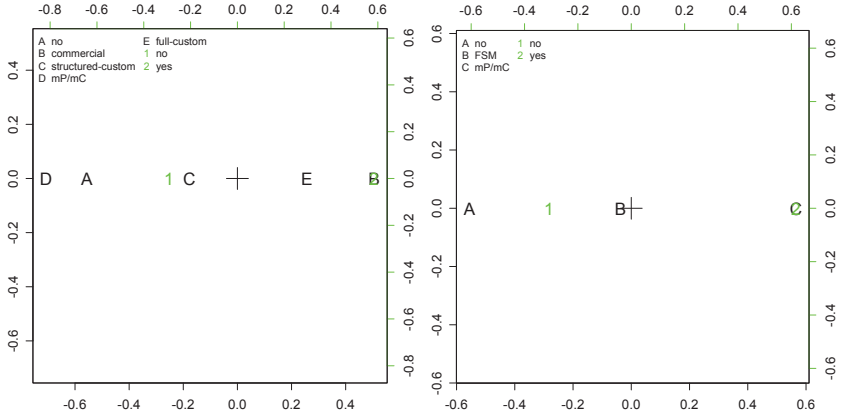
Question 2.33 *Does the chosen implant* PCC design *and* PCC architecture *have an impact on its* reliability *provisions?*

It is very interesting to see how reliability-enabled devices are distributed among the various PCC-design and PCC-architecture types encountered so far. To do so, we first need to establish that there is indeed a correlation between 'reliability' and each of these variables. We perform chi-square tests and they both return significant p-values at the .05 level (X^2 = 8.1867 and p = 0.08497 for PCC design; X^2 = 9.1477 and p = 0.0103 for PCC architecture). These correlations have been visually illustrated in the bar-charts of Figures 2.45a and 2.45b. Respective CA plots are also provided in Figure 2.45c and 2.45d to explore these correlations also visually.

it is further extracted that more *radical* reliability provisions are assumed by full-custom designs as compared to commercial ones. For the former, designers have actively gone out of their way to incorporate testable devices (*class 6*), with various diagnostic and testing circuits (*classes 2* and *4*) as well as safety and duplicate circuits (*classes 1* and *3*). For the latter, designers have appeared to rely mostly on built-in capabilities of the commercial PCCs they are using. For instance, the otherwise unutilized on-chip ADC of a commercial μC is employed as a feedback element which monitors the implant battery level and temperature and informs the user or takes some other course of action.



(a) *Reliability w.r.t. PCC design.* ***(b)*** *Reliability w.r.t. PCC architecture.*



(c) *CA plot with PCC design.* ***(d)*** *CA plot with PCC architecture.*

Figure 2.45: *Distribution of reliability per PCC design and per PCC architecture.*

The general impression formed is that in commercial PCC-based systems, designers have treated reliability more as a secondary design goal and less as a primary issue and have, thus, supported it only through unutilized resources of the commercial components at hand. The reason for such an attitude might be sought in the fact that commercial components have more or less known MTTF (mean time to failure) and pre-verified, proven functionality. Thus, in such cases designers seem not to feel further design for reliability is needed. There could be a more practical aspect to this attitude. Namely, "injecting" reliability schemes into commercial cores is more difficult than into custom ones (where one is in full control of the design), which also explains the less radical approaches taken for the former.

For structured-custom and semi-custom designs, the number of available cases

is not sufficient to extract safe results; still, in these cases one can anticipate the low percentage of reliable designs, as well. Devices with the overall lowest percentage (1 out of 14 cases, 7%) are those not including a PCC. This could be justified by the fact that such systems are usually hardwired implementations with highly predictable, straightforward behavior either due to their low design complexity or due to their static behavior. Thus, their functionality is easier to test and their failure rate easier to predict.

Figure 2.45b, reveals another aspect of the reliability results. The "no PCC" column is the same as that in Figure 2.45a. The other columns report on the percentage of "reliable" implants based on FSM cores and on μC/μP cores. μC- and μP-based systems present the highest percentage of reliability provisions (60%) while FSM-based ones follow with roughly half that (32%). This large gap between the two PCC architecture styles can be explained by the fact that while FSM-based cores are hardware implementations of the implant control logic, μC/μP cores constitute software implementations of the control unit. Contrary to a state machine, code execution offers more flexibility to a system (see also previous discussion on programmability) but it also bears more complex interactions and, in effect, more unpredictable behavior inside the system and its sub-blocks. This also explains further the extremely low percentage of reliability for implants with no PCC: a coreless design is certainly less complex (thus, more predictable) than a FSM-based or a μC/μP-based one.

There is a second reason why FSM cores score lower than μCs and μPs in terms of reliability provisions and it has to do with system complexity as well. μCs and μPs are more complex and, thus, require higher reliability but at the same time can provide this extra reliability at a lower cost than FSMs. Imagine the above mentioned ADC which monitors the battery voltage level and temperature and reports it back to the PCC. If the PCC is based on software execution, the overhead in design time and resources (CPU load and code/memory size) is negligible. The expansion is as simple as adding a separate (diagnostic) process to the executed code. If, on the other hand, such kind of control needs to be implemented in a FSM, new (diagnostic) machine states are needed which probably also need to run in parallel with the main FSM cycle. This incurs non-trivial costs in design time but also in system resources, e.g. more transistors, higher power consumption, additional pins (connections between the FSM and the battery) and so on.

Conclusively, μC/μP-based PCCs facilitate but also accommodate a higher degree of reliability, in the general case. The vigilant reader will, of course, realize that the above conclusion is nothing more than the comparison be-

tween hardware-based and software-based implementations of reliability. It is rather obvious that if hardware-based reliability is desired also for the cases of μC/μP-based PCCs, that is, if reliability features are to be supported by dedicated hardware instead of software modules running inside the core, extra effort and transistors are needed. Still, results indicate that software-based techniques have been mostly preferred for μC/μP-based implants, so far. As a last remark, reliability provisions should be enhanced in future implant designs, firstly, due to the mission-critical (medical) purpose they serve. Secondly, increasing design complexity under scaling technology makes hardware components more amenable to faults, thus, demanding design for reliability.

2.7 Summary

In this chapter we have presented a taxonomy and in-depth analysis of a large number of implantable systems covering mostly the 12-year-long period 1994–2005. A large number of different device attributes have been investigated, resulting in a rather detailed classification (although some implant aspects have been consciously omitted or constrained). The data has been concentrated in tables and, based on them, an involved commentary of the findings has been presented. The survey has yielded interesting and, at times, counter-intuitive results. The fact that modern microelectronic implants can be effectively grouped in only two main categories in terms of functionality, with similar power and other requirements, the net increase in the dynamic power consumption of implants and the drop in reliability provisions over the years are only a few of these results. The lessons learned throughout this survey has provided us with sufficient background knowledge and insights to confidently take the next step which is the conceptualization of the SiMS project.

Note. The content of this chapter is based on the the following papers:

C. Strydis, **Biomedical microelectronic implants**, MSc Thesis, 2005.

C. Strydis, G.N. Gaydadjiev, S. Vassiliadis, **Implantable Microelectronic Devices: A Comprehensive Review**, Technical Report (CE-TR-2006-01), 2006.

3

The SiMS concept & background

THE modern world of rapid socioeconomic changes and technological leaps has created an opportune environment for implantable systems to evolve into much more than applications of pure academic interest. It has been an odd 50-year-long journey from the general-public scepticism towards implants to their current status as undebatable solutions against certain pathoses, such as heart arrhythmia, PD, deafness and more. The fully implantable pacemaker, developed in 1958-59 by Wilson Greatbatch and William M. Chardack, has been the first device to be implanted successfully into the human body and to operate seamlessly for long periods of time. More importantly, this device has acted as the catalyst on the general public closed-mindedness against biomedical implants. Indicative of the penetration and impact pacemakers have achieved is the fact that, in the U.S. alone, a total number of 180,000 implantable pacemakers have been registered for the year 2005 (source: American Heart Association [35]).

Their becoming commodity products, implants are nowadays doing more than simply follow societal trends. They are becoming an influential factor in healthcare and – eventually, we believe – public-policy making. The phenomenon has been witnessed before, with other life-altering technological advances such as mobile telephony. These complex (direct-inverse) socioeconomic and technical (as discussed in the previous chapter) relations have been the major incentive for the present thesis work and have led to our proposing of a new paradigm for implantable devices known as **SiMS – Smart implantable Medical Systems**. Building on the technological trends pinpointed in Chapter 2, in the present chapter we will concisely discuss the *socioeconomic drifts* necessitating the inception of SiMS. We will, then, describe the *SiMS concept* in detail and present the *background information* needed to realize it.

3.1 Motivating a new generation of implants

The research work included in this thesis work has initially been stimulated by a number of observations in the socioeconomic as well as in the technological plane. By performing an in-depth survey of the implant field (seen in Chapter 2), and through the process of this research, we became increasingly knowledgeable in the field. Hands-on lessons learned, along with the generally observed societal trends, have helped in conceiving and refining the SiMS framework; that is, a new approach in designing microelectronic implants. In the following section, we briefly review the observations motivating SiMS.

3.1.1 Socioeconomic trends

In the face of ongoing socioeconomic advances, healthcare in the 21st century is changing rapidly. In advanced countries, in particular, healthcare is moving from a public to a more personalized nature [3, 26]. In advanced countries the following cascading trends are currently being observed:

- Population is aging through a net reduction in birth rates combined with an increase in life expectancy;
- Healthcare costs are growing out of proportion; and
- Higher demands for betterment of quality of life are placed (health, fitness, convenience etc.).

The costs of healthcare worldwide are increasing every year. In *the Netherlands*, in particular, the government is trying to keep the health insurances affordable for all citizens by periodically reorganizing the system. Since healthcare spending always increases at a much faster rate than the average income, such practices work only for a limited period of time. The rising healthcare costs, in combination with population aging (i.e. more potential customers for the healthcare system), form a tough challenge for modern societies.

Presently observed cost overruns and inefficiencies are clear indications of systemic failures in the existent healthcare construct. To cope with such phenomena, many contemporary healthcare systems have set off implementing the New-Public-Management (NPM) paradigm. This paradigm, in the words of Kickert, claims that:

> "Under conditions of heavy public demands but a severely constrained public budget, the only feasible alternative to cutting public services or raising taxes, seems to be to reduce costs, increase effectiveness and efficiency, and deliver 'more value for the money'."

The paradigm has already received strong criticism [22, 74], yet is it a reality pushing the public sector to become more businesslike, 'work better and cost less', and become more client-oriented. In the *legal domain*, governmental parties in many countries are now attempting to preempt the coming change by revising the standing legislation and passing new one in order to cope with this new era [102].

Such socioeconomic trends have given birth to the notion of *personalized healthcare*. The term introduces a new approach to effective healthcare – as far as economics go, at least – whereby default hospitalization and generic treatment of patients is discouraged and supplanted by *patient-specific* prognosis, diagnosis and, mainly, treatment. It goes without saying that *technology* will be the vehicle for enabling personalized healthcare; similar trends have already been witnessed in the cell-phone and portable-computing revolutions.

Better use of technology – and, in our case, implants – can and should be used to get control of healthcare costs. For example, continuous monitoring of physiological parameters can be used instead of occasional meetings with the doctor. Having an up-to-date and complete picture of the changes in a patients condition will enable disease prognosis, which by definition is more effective and less costly than disease treatment. It should be stressed that such technology will be used not only for high-risk or chronic patients, but also for general lower-risk patients over periods of normal activity in their home or work environment.

3.1.2 Technological trends

A number of technological innovations is attempting to carry healthcare systems to the next level, such as wearable electronics, portable medical monitors, body-area networks (BANs) and, last but not least, microelectronic implants. Implants have been around for more than 50 years, yet over the last decade they have being designed for an expanding range of applications ranging from implantable microstimulators to pervasive, in-vivo monitoring and data-logging devices, as detailed in Section 2.6.1.

Implants have clearly benefitted from the astounding recent *technology-miniaturization* trends [66], boasting smaller sizes, lower power consumption and increased performance of the transistor devices. Simply put, while the human-body dimensions have not changed, microelectronics dimensions have – by proportion – shrank to such an extent that modern implants are becoming sufficiently sophisticated and small so as to treat various human pathoses, even at the most constrained parts of the human body. It is this practical property along with their wider acceptance in modern societies that is making them a primary technological driver towards personalized healthcare.

3.1.3 Survey-based implant trends

The previous discussion gives indications that current socioeconomic and technological trends are in place to favor the wide deployment of microelectronic implants. Yet, the survey performed in Chapter 2 has revealed a shift in the implant-design paradigm: implant PCCs are gradually moving from custom-designed (ASIC), application-specific (e.g. FSM-based) systems [50, 96, 106] to more off-the-shelf and generic (e.g. $\mu P/\mu C$-based) ones [25, 78, 84]. Furthermore, PCC design is becoming more streamlined and structured than it used to be and that, in the near future, implant functionality will be based on executed software (written in some high-level, established language like C) rather than on hardwired circuits.

Such a transition has been previously witnessed in other computer-engineering niches (a prominent example being the Personal Computer – PC) and is anticipated to lead to a booming in implant designs and applications. As desirable as such a turn of events is, alas, it does not come for free. An adverse effect is that reported implant *power budgets are increasing* over time, even though transistor dimensions are shrinking and implemented device functionality is not overtly complex.

Another revealed pitfall is the serious *absence of design for reliability* (DFR) in implants: software-based, ad-hoc reliability techniques have been replacing inherently reliable implant designs over the years. For a field of highly-mission critical embedded systems where human lives and high costs are involved, this poses a significant problem.

Last but not least, even though implant designs are becoming more structured, product development is still *highly application-specific*. Already established product cases such as the family of pacemakers introduced by Medtronic, Inc. [91], where previous design expertise is (re)used to enhance the next de-

vice version, are currently the exception[1]. Reasons for this long-lasting attitude are thought to be the high-risk and, thus, high-cost characteristics of the implant market which, understandably, force companies to assume a highly conservative (and often secretive) stance towards new product development. Unfortunately, it is this particular attitude that (a) is keeping implant-device costs prohibitively high for a large part of the population and (b) is limiting the gamut of potential implant applications.

To sum up, currently witnessed implant-design trends weave a bright picture for future implants which is only hindered by certain existing issues; namely, rising power consumption, lack of DFR and recurring (re)design costs.

3.2 Smart implantable Medical Systems (SiMS)

As the previous sections have elucidated, microelectronic implants are one of the primary vehicles for personalized healthcare in modern societies. Nevertheless, the current status-quo in the field suffers from certain problems which have to be dealt with, for the first time, in a top-down fashion. Enter **SiMS**.

3.2.1 The SiMS concept

The main goal of this thesis is to deliver **SiMS**[2]; that is, a *systematic approach* (thus, a *framework*) which provides biomedical researchers with a toolbox of ready-to-use, highly reliable implant sub-systems and models in order to construct (optimal) implants for various medical applications. The SiMS framework has to guarantee the following essential attributes:

- high dependability (reliability, availability, maintainability and safety),
- modular, versatile design for design reuse,
- ultra-low power (ULP) consumption, and
- miniature size.

Devices built on the SiMS framework will be small, *fine-tuned* implantable devices to the application at hand, yet built of *generic* components. Without

[1] Notice that, even in this case, we have a succession story of incremental pacemaker/defibrillator designs which still target a very narrow application field.

[2] Official SiMS website: `http://sims.et.tudelft.nl`

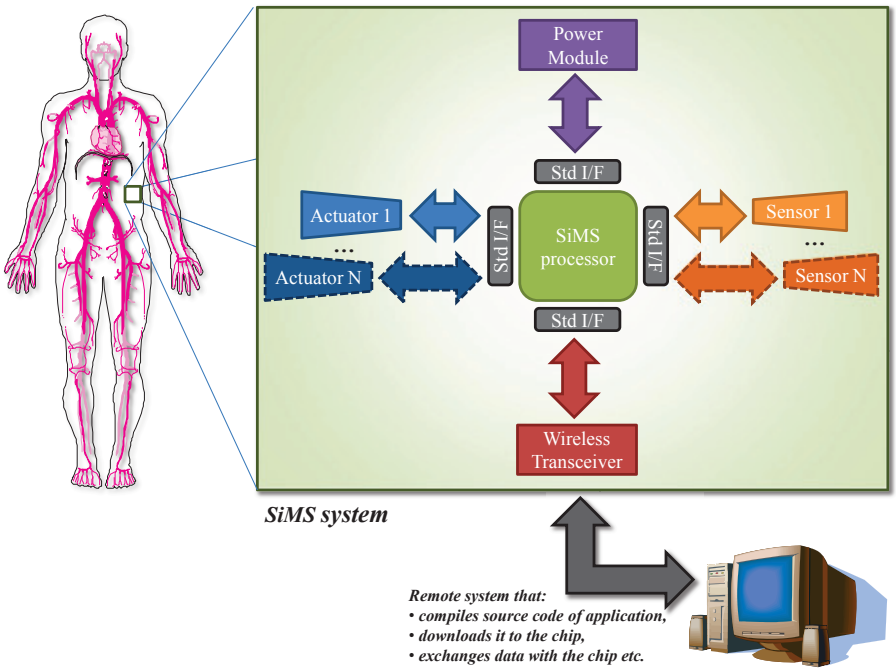


Figure 3.1: *Block diagram of the SiMS concept.*

requiring redesign, they are able to *measure* and/or *regulate* one or multiple biomedical parameters simultaneously and communicate with external (out-of-body) computing equipment wirelessly. Given that such devices are directly related to human life, they will be characterized by very high **reliability** and some degree of **autonomy** and **self-awareness** (within extremely demanding **low power** and **size** constraints). Within the SiMS context, performance does not need to be particularly high. As the survey in Chapter 2 has revealed, high PCC frequencies are not needed. Rough frequencies of 4 *MHz* are deemed more than enough to execute the implant tasks within real-time deadlines set by the applications. Of course, we expect slightly higher (maximal) frequencies for the SiMS devices as a necessary side-effect of the generic nature we impose on SiMS.

Our long-term goal is silicon, multi-sensor/-actuator, single-chip, wireless medical systems. Such systems will be produced using fully integrated CMOS processes. In addition, they will be capable of **context-sensitive behavior** (thus, smart), due to their multi-parameter awareness and communication abilities. The combination of the aforementioned issues with the envisioned **mod-**

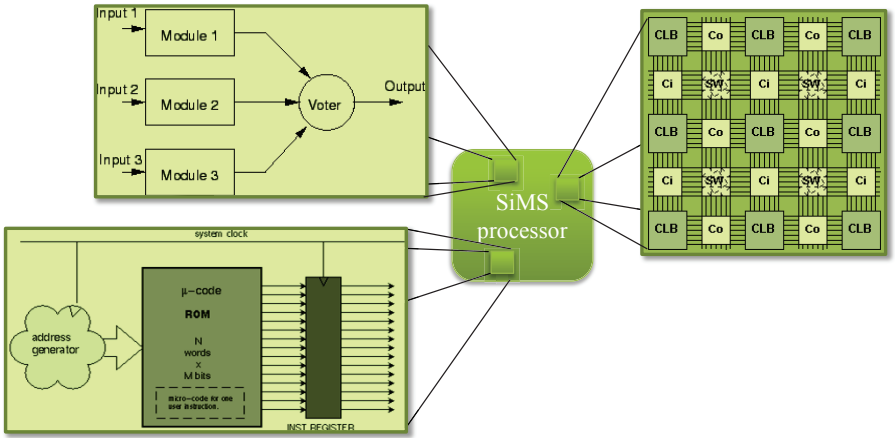


Figure 3.2: *Reconfigurable hardware, replicated hardware and microcoded units in the SiMS processor.*

ular system approach, introduces new research challenges. A conceptual diagram of the SiMS framework is depicted in Figure 3.1. In the next sections we give a short description of the various SiMS subsystems.

3.2.2 SiMS digital processor

The digital processor (the so-called **SiMS processor**) forms the main processing/controlling unit characterized by extremely low power consumption and fault tolerance. The SiMS processor (essentially the PCC of a SiMS implant) is responsible for:

- collecting and processing data from the sensors and/or regulating the functionality of the actuators,
- forwarding data to the radio transceiver for telemetering information externally and for accepting commands received from the external host (e.g. the treating doctors computer),
- controlling the functionality of the various implant subsystems; for instance, it is able to turn subsystems off when idling for long periods of time and back on when their operation is required.

Implantable devices are liable to a set of strict (often extreme) specifications due to the sensitive and demanding nature of their living "environment" and

SiMS have to conform to them. In this context, the SiMS processor will be crucial in delivering highly dependable implantable devices. In principle, it will achieve this by displaying attributes of error detection and error correction as well as self-testing and self-repairing properties. Such features will be achieved by redundant hardware structures (e.g. replicated modules) that will continuously check, correct and/or isolate faulty modules. To this end, reconfigurable standard cells are also considered, as shown in Figure 3.2. If reconfigurable hardware proves to be to expensive power-wise, incorporation of microcoded units is also considered as a trade-off between power consumption and chip size. Since performance is not a primary concern in SiMS, microcode may turn out to be an interesting (and, perhaps, more reliable) alternative for implants. In addition to the hardware, all SiMS software modules have to be error-resistant and self-correcting, as well (e.g. unique checksums shall be assigned to the instructions for picking out soft errors). The above features will give implantable systems the much-desired properties of robustness and safety, which are rudimentary ones, given the medical nature of the applications.

The SiMS processor will be small (16-bit architecture or less) featuring only a few thousand transistor devices at a maximum. The obvious reasons for this choice are the small-size and the ULP constraints ($100 - \mu W$ order of magnitude or lower) that implantable devices have to adhere to. A slightly larger size may deem a device unusable in the case of e.g. intra-cortical implantation whereas high power consumption: a) drains the battery of an implantable device rapidly, and b) causes potential damage (e.g. burns) to its surrounding tissue due to heat dissipation. A less obvious but equally important reason for a small architecture is that it allows achieving and maintaining the reliability of the system more easily.

Finally, the SiMS processor will define specific *interfaces* to all peripheral modules (biosensor(s), bioactuator(s), communication module, power module) and will, thus, standardize and simplify the way a large (infinite, for practical applications) number of different modules are selected for different applications. In this way, SiMS devices will become multi-featured, multi-capable systems easily built to the specifications of diverse applications by plugging-in standardized, pre-verified, pretested peripheral cores to the implant PCC (i.e. SiMS processor).

3.2.3 Typical SiMS workloads

With the exception, perhaps, of the implantable pacemaker and the cochlear implant – the former expect QRS complexes from the heart, the second human-

audible sounds – there has never been a commonly agreed upon list of "typical implant" **workloads** (also known as benchmarks[3]). One reason is the prohibitive diversity of the medical applications serviced by modern implants; another could be the fact that, before now, there has never been proposed a generic processor for implants (like the SiMS processor), thus creating the need for an established list of processor workloads.

Our performed survey and experience so far indicates that there can indeed be identified such a list of workloads which will be commonly encountered in future implantable systems. This will serve a two goals: First, a common base for comparisons among different implant designs can be established, similarly to the benchmarks driving PC comparisons. Second, the benchmarks are indispensable in drawing the specifications of the SiMS processor. In the next chapter, we will take a closer look at what we consider a representative mix of implant workloads, such as compression, encryption and data-integrity algorithms.

3.2.4 SiMS HLL Compiler

The compiler will be responsible for generating the machine code to be executed by the SiMS architecture. Application design will be straightforward: a desired application behavior, defined by e.g. a doctor, will be properly encoded in a high-level language (HLL) which will, then, be compiled to machine code and directly mapped to the Instruction-Set Architecture (ISA) defined during the first work package of this project.

Like all standard compilers, the compiler tied to the SiMS platform will be able to perform *code optimizations* (instruction scheduling etc.). In so doing, the instruction count (and, thus, execution time) of specific applications may be reduced allowing for lower power consumption and, therefore, for prolonged implantable device lifetime (i.e. autonomy). Other, advanced compiler issues like *reconfigurable- or replicated-resource allocation* at static and/or dynamic time will also be addressed. This, especially since optimizations have to be performed not only for performance but also for *small memory footprint*, *low power consumption* and *fault tolerance*.

The crucial issue of dependability will be indeed treated in the compiler level, too. The compiler will actively handle this aspect by also accepting *application-specific constraint files* along with the main *application source*

[3]Following the established computer-engineering jargon, the terms "workload" and "benchmark" are equivalent and will be used interchangeably in this thesis without further explanation.

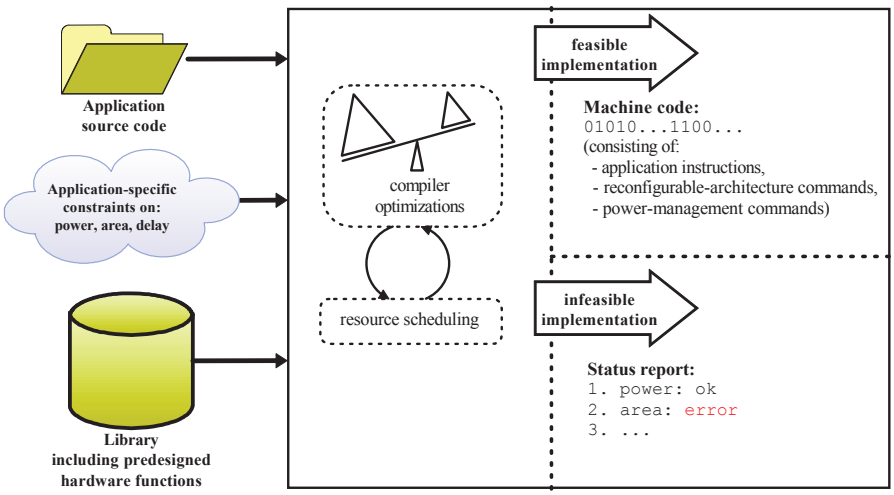


Figure 3.3: *Overview of dependability guarantee through SiMS-compiler provisions.*

code. Such a file will include application-specific information regarding, for instance, the nominal and maximal power consumption allowed, the area utilization and the processing speed of the targeted application (see Figure 3.3). This set of constraints will - per case - reflect general specifications of the application. For example, similar code transformations may lead to different results in relation to the context – where, inside the human body, the targeted system will be implanted. As an example, the same power dissipation that is prohibitively large for usage inside the brain may be tolerated in some other cases. Given the source code and the set of constraints, the compiler will then determine if a realistic solution on the SiMS platform exists. It is an extra safety precaution that, if such solution does not exist, the compiler will generate an error report and will not output any machine code for the device. In case (some of) the constraints are not met, changes in the compilation switches could also be suggested, e.g. decreasing the level of software-based fault tolerance will result in smaller fault coverage but perhaps also less power, less memory utilization etc..

The interplay between the SiMS compiler and architecture can be the design spot where ample research and interesting ideas can be generated. Except for *safe compilation* and fault tolerance at the software-only or hardware-only levels, solutions that require both the participation of the compiler and the architecture are also considered. An example of such a collaborative approach

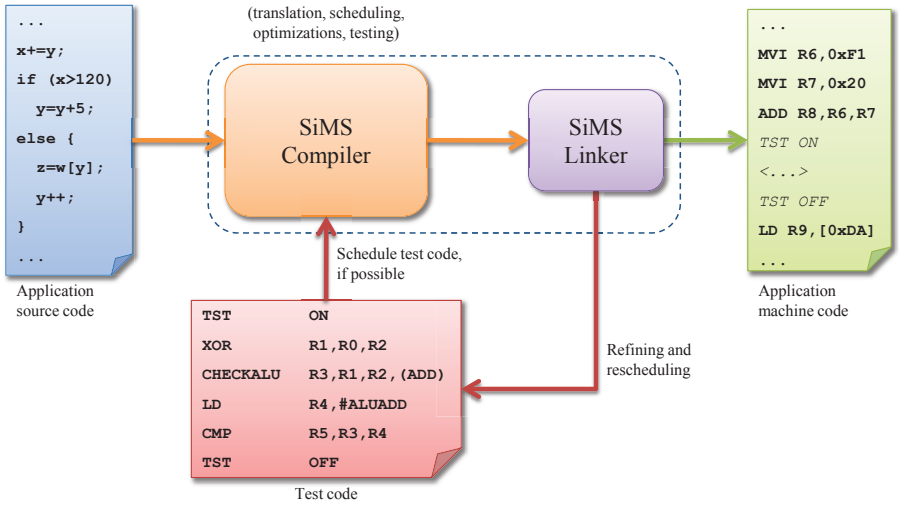


Figure 3.4: *Compiler scheduling of test instructions.*

to fault tolerance can be the following: (a) develop a set of test instructions for the critical parts of the architecture, and (b) have the compiler schedule such instructions efficiently (e.g. replace some NOP instructions with test instructions) for execution while respecting performance, power and program-size constraints given by the application addressed. This is illustrated in Figure 3.4.

3.2.5 SiMS peripherals

Biosensors and bioactuators attached to and under direct control of the SiMS implants are considered in this part. Investigation of new sensor and actuator peripheral modules or improving of mature ones, all well-suited for the biomedical domain, is needed. A typical improvement example is an implantable glucose detector. Such modules have commonly been based on transducing elements that unfortunately age (i.e. their performance deteriorates with time). To make matters worse, the body reacts to the exposed sensor/actuator front-ends dulling their accuracy and sensitivity. Potential work could be done on a glucose detector based on optical technology, so as to avoid chemical interaction with the living medium and, in effect, performance degradation.

The endmost goal for SiMS peripherals is proving the modularity benefits of the SiMS-platform approach through designing new sensing and actuating elements with a *standard, common interface* to the SiMS processor (see also discussion below on “SiMS chip interconnects”). Furthermore, much effort

will be put on the transducing system of those sensory and actuating elements in an attempt to boost *sensitivity* and *performance*. By improving peripheral-module design, *size* and *power consumption* will directly benefit.

To a further extent, limited yet built-in self-tests will be supported (in an autonomous or processor-assisted fashion) by these modules offering increased *reliability* and *safety*. On another level, the micromachining implementation method to be used will mainly affect size issues (micron and submicron technology) but also power issues. Lastly, an important topic is that, unlike the processor which is isolated from the environment, sensors and actuators may be in direct physical contact with the living tissue. Therefore, additional issues of *biocompatibility* of the materials used are raised and are to be resolved in this part.

3.2.6 SiMS wireless transceiver

The wireless transceiver module is the communication unit of the SiMS device. It is responsible for: transmitting and receiving various types of information, viz. (medical) sensory information, control information, status information, and providing wireless connectivity on demand and with a guaranteed QoS, even in radio-harsh environments.

The design of the SiMS wireless transceiver module is accompanied by significant challenges at the system, circuit and technology level: being part of an implantable device, the wireless transceiver module, including its antenna, must have a small form factor. Being part of an implantable device with a small form factor, reliable wireless connectivity needs to be established and maintained with low power consumption to ensure long battery life and/or to allow the use of alternative power supplies that convert electromagnetic, chemical, thermal or other type of energy into electrical energy. Reliable wireless connectivity, even in radio-harsh environments, requires the use of multiple frequency bands, multiple standards, possibly including ultra-wideband (UWB) techniques. Main circuit design challenges for the full integration of multi-band, multi-standard and UWB devices include on-chip image rejection, carrier/pulse detection and generation and provision of sufficient dynamic range.

Whereas the transceiver circuitry determines the instantaneous power consumption of the transceiver module, the average consumption depends on the power management of the complete SiMS system. This implies that not only local, but also global (at all layers and at all time) power optimization and awareness are important for extending the lifetime of the implantable device.

Power management for mobile devices has already become one of the fastest growing segments in wireless IC revenue. Setting the performance by means of adaptive circuitry is a way to manage power consumption in the RF and baseband parts of the transceiver, while at the same time satisfying the various functional requirements outlined above.

The UWB technique[4] is one of the prime candidates for implementing into the wireless transceiver due to its natural robustness to noise and support for high data transfer rates. For instance, UWB with high-low frequencies of 5 *GHz* – 400 *MHz* can sustain a nominal rate of 1 *Gbps*. Even though a range of kbps shall be sufficient for SiMS applications, this nominal rate can be reduced, i.e. traded off for higher communication robustness and/or lower power consumption. To exemplify, through careful use of UWB techniques, data transfer could be achieved at 5 to 10 *mW* per *Mbit* of information.

Designing the wireless transceiver shall also include an extensive study of the attenuation the information signal undergoes when cross varying medium compositions (e.g. soft/hard tissue, body fluids etc.). The idea is to build a module which is adjustable to different application scenarios by, for instance, changing its resonant frequencies or the throughput, depending on the setup.

3.2.7 SiMS chip interfaces

In order to deliver the desired SiMS modularity, interoperability and reusability of the SiMS components, *standard interfaces* have to be researched and developed. All sub-blocks of the SiMS framework have to adhere to the same interfaces so that donning and doffing of different components at design time is possible. These standard interfaces have to be compact, low-power consuming, reliable and – within specific margins – fast. Some work on the field of interfaces has already been laid out in the field of implants [41] as well as wireless sensor-networks (WSNs) [149], yet to serve the purposes of SiMS, account of all above attributes have to be taken into account when designing a new interface protocol and architecture.

[4]The UWB approach and all related design and implementation concepts discussed here originate from interactions with SiMS partner Prof. Wouter A. Serdijn, ELCA Laboratory, EEMCS Department, TU Delft.

3.2.8 Miscellaneous SiMS components

In order for a completely functional SiMS device to be produced, there is a plethora of aspects to be resolved first. So far, we have mentioned the prominent aspects (with respect to our expertise and available time frame). However, there is a number of other, equally significant issues which need to be dealt with before a complete, commercial system can be delivered to the market. We mention these aspects only briefly:

- Antenna: On-chip, miniature-size, high-gain antennas;
- Power source: New chemical batteries with significantly reduced geometries, longer lifespan and predictable, safe behavior need to be developed for powering the SiMS systems. Power scavenging is another explored alternative for SiMS;
- Standardized interfaces between the various subsystems;
- Design for Electro-Magnetic Compliance (EMC);
- Heat dissipation;
- Device packaging;
- Information security.

3.2.9 SiMS relevance

After going through the various SiMS building blocks, one point must be made clear. Within the SiMS project, it is *not* our express goal to propose novel implant applications but, rather, to specify a sound framework upon which many existent (but certainly not all) and new implant applications can be built. Outright, envisioned benefits of our approach are mainly threefold:

- radical shrinking of development times and costs;
- clear separation of the expertise for building the various SiMS subsystems and the final SiMS implants; and
- drastic increase of overall device reliability.

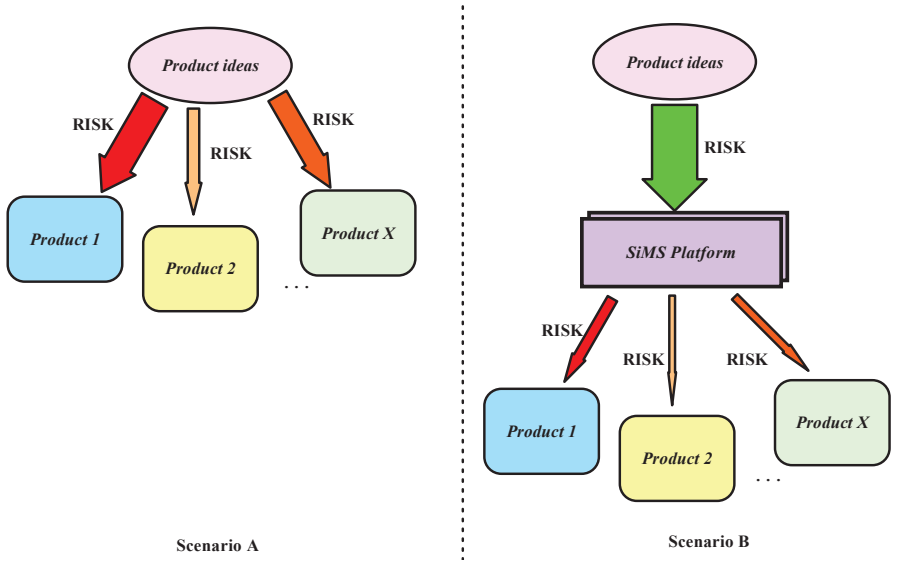


Figure 3.5: *Involved risk factors with (A) and without (B) a design platform. (The thickness of the arrows is proportional to the risk weight).*

SiMS shall guarantee a reduction of development times by providing a solid substrate onto which prior art will be brought together, combined and integrated in the final product. Such prior art will be in the form of Intellectual-Property (IP) hardware and software components, all proven, pre-verified and pre-tested according to (inter)national medical-safety regulations. This shall, in turn, guarantee an increase in component- and device-level reliability.

By being fully aware that implantable devices are fruits of a multidisciplinary, combined effort, we also work within the SiMS framework towards a clear separation of partner expertices. That is, we aim at a framework where various types of engineers provide the system architecture, the sensors and actuators, the power source, the wireless transceiver etc. while medical experts are actively involved in composing, testing and fine-tuning the final system to the particular patient needs.

3.2.10 Minimizing risks and costs

The proposed SiMS modular approach will decrease the time between idea and product and balance the risks involved in the development process. In addition, due to the multidisciplinary nature of such systems, there are still

many unresolved integration issues, e.g. not always the combination of the individual best practices will lead to the best system. By addressing these issues and proposing pre-tested solutions, we provide the industrial partners with an operational platform that they can use for subsequent commercial exploitation.

Even though highest priority in medical and, specifically, implantable devices is not usually attributed to the device cost, it is still a boundary condition that can make the difference in the market. As graphically depicted in Figure 3.5, the currently existing approach (A) entails high risks for the development of every new device family, whereas the proposed approach (B) involves a one-time high-risk and -effort step during the platform design and considerably lower risks for the products deployed on it thereafter. The term "risk", includes various risk parameters and challenges appearing during the design, the development and also during the normal operation phase of an implantable system. A platform-based approach for system synthesis (such as SiMS is) will allow for dramatically lower development times and costs of implantable systems. Also importantly, it will provide a more standardized way of building new medical devices. In so doing, it will further underpin the reliability issue.

3.2.11 Prior art on generic implant designs

In the past, a few attempts have been made to design generic implants with a certain degree of modularity in order to make them capable of adapting to different application scenarios. These cases have already been incorporated in the implant-survey findings presented in Chapter 2. Since they are considered to the closest cases to the SiMS approach, they are briefly described below.

Fernald et al. [41, 42] have proposed a modular microprocessor architecture which accepts various peripheral modules such as sensors, actuators and transceivers. Application flexibility is underpinned by a dual ring-bus interconnect linking an arbitrary number of modules to the processing core which is a fully featured 16-bit μP (PERC), based on Hector [93]. Command and data packets, traveling across each bus, have predefined, consistent structures and plugged modules are built to interface to them.

Contrary to the additive nature of the above design, Smith et al. [112,127] have addressed the problem of flexibility from a subtractive angle. An implantable stimulator device with provisions for a large set of peripherals was designed. Given a specific application, unutilized components of the initial, baseline design can be removed, resulting in a reduced system, tailored to the application needs and with lower power/area requirements than those of the base design.

Valdastri et al. [140] have presented a versatile implantable platform that provides multi-channel telemetry of measured biosignals. Its versatility resides in its ability to support different types of sensors and to allow for easy reprogramming so as to fulfill different application requirements. To demonstrate the correctness of the concept, a specific case study is implemented for gastric-pressure monitoring which is a PCB-mounted assembly, supporting up to 3 sensor channels. This implant can transmit digitally modulated data to an external receiver over a wireless link with robust error control.

Furthermore, Salmons et al. [117] perform a design and comparative study between an ASIC-based and a microcontroller-based microstimulator device for restoring functionality to paralyzed muscles. Analysis has shown that, if carefully designed with low-power modes and checked for software bugs, the latter version is beneficial to the ASIC with respect to development and testing costs.

Perhaps the closest work to SiMS is that conducted by Fernald et al. SiMS is original in that it attempts to specify (among other components) a truly generic, low-power and fault-tolerant processor architecture while at the same time providing the performance needed by current and future applications in the field. The effort in this document henceforth is to detail the steps taken to explore the specifications of such a processor architecture.

3.3 Technical background

The SiMS concept, as outlined in the previous sections, requires (re)touching all aspects of implant design – from the application source code all the way down to the gate level (or even lower for the peripherals). While this provides a wide scope of potential research themes to pursue, our particular expertise within the Computer Engineering laboratory as well as our available time frame forces us to focus on a a few aspects of SiMS, for the purposes of this thesis work.

We believe that most serious (and original) groundwork needs to be laid in the PCC of SiMS; that is, in the digital processor (and assorted HLL compiler). Since the processor has to be designed before its matching compiler, the primary focus in this thesis and the topic of discussion henceforth is the SiMS processor[5]. **We advocate the design of an ULP and generic processor**

[5]Concurrently, PhD research work on the SiMS compiler is being conducted within the Computer Engineering Laboratory by Ghazaleh Nazarian, Ir..

for implants with explicit provisions for fault-tolerant operation and suitable for covering a large subset of current and, most importantly, future implant applications. Although fault tolerance has been one of our primary goals for this thesis work, it proved to be rather difficult to utilize existing tools or build our own ones to model and explore it within the SiMS processor; at least, as a first step. We will revisit this issue in Chapter 6 when we focus on processor DSE tools.

In order to specify and design such a processor we, first, had to set up a suitable *evaluation environment* along with representative implant*workloads* for conducting our profiling studies. Figuring out what "representative" is in the context of biomedical implants and selecting the right evaluation environment has proven to be a significant challenge in itself since prior art (literature and tools) in the field is seriously absent. We have, therefore, made our choices based on: (a) availability/accessibility of resources (source code, simulators, tools etc.), (b) educated hints taken from the implant survey we have already performed in Chapter 2, and (c) our intuition, running experience with implantable devices and past experience with embedded systems, in general.

3.3.1 Work organization

The first step was picking a suitable simulator for our SiMS processor. We, then, selected a list of benchmark categories (e.g. compression benchmarks, encryption benchmarks) that we considered representative of workloads to be executed in actual, future implants. For each category, we identified the best-performing candidates under strict performance, power, size and other restrictions. The next step was to group these and some additional benchmarks into a novel benchmark suite for implant processor. With all the above pieces in place, we were able to explore optimal configurations for some of the microarchitectural features of the SiMS processor, as allowed by the available simulation environment. Finally, with all tools in place, lessons learned in the process and some enhancements to the existing benchmark suite, we expanded our tools and proceeded in building an automated Design-Space Exploration (DSE) tool for optimal SiMS processors.

For each of the above outlined steps, the current state of the art – that is, the technical background – has been studied. Following the above task organization, the upcoming sections aggregate and present the existing prior art.

3.3.2 Processor simulators

In order to explore features of the SiMS processor, we have taken the approach of, first, modeling the processor and profiling its behavior in a suitable simulation environment and, then, transferring this information to a hardware design tool. We would ideally want a processor simulation environment with the following features:

- a **flexible simulation environment** where different processor modules could be tweaked, added or completely removed;
- **accurate modeling** of various architectural (e.g. cycle-accurate instruction execution) and microarchitectural (e.g. cache geometries) parameters;
- the ability to run workloads written in some popular HLL language (e.g. C), meaning the processor simulator should come with **compiler support** – and complete binary utilities, for that matter;
- support for **error injection** and **hardware fault models**;
- the ability to output – except for the correct program output – additional **metadata** for its various subsystems; that is, performance, power, area and failure-rate figures, to name a few.

Coming up with a processor simulator supporting the above features is a non-trivial exercise in simulation tools in its own accord and a topic of serious research for many years now. Accurate modeling of power consumption within a simulation environment is already extremely complex. Modeling of faulty behavior is even less mature in simulation and EDA environments at the moment. To make matters worse, some of the features in the above "wish list" represent conflicting simulator-design requirements: For instance, the more flexible a simulator is in its parameters, the wider the exploration capabilities it offers but, also, the less accurate the modeled processor can be. This is easily observable in the power models of existing simulators. The more generic the modeled system is, the less precise (and deterministic) the power estimation can be.

Last but not least, in all the above we should also add the obvious requirement that the simulator should be modeling devices in the same or similar application field as the targeted systems – in this case, implantable devices. For

instance, simulators of processors with complex features such as wide instruction issue, out-of-order execution, multithreading, multicore implementations and so on would be completely overshooting our desired ULP, miniature SiMS processor concept.

We have investigated suitable and, more importantly, publicly available simulators and – to verify our previous claims – we have come up with an almost empty list. Eventually, through our contacts[6] we have come across XTREM [24], a modified version of SimpleScalar/ARM [8, 18]. The XTREM simulator is a **cycle-accurate**, **microarchitectural**, **performance-** and **power-simulator** for the high-performance, low-power Intel XScale core [65]. It models the effective switching-node capacitance of various functional units inside the core, following a similar modeling methodology to the one found in Wattch [17]. XTREM has been selected for its straight-forward functionality but mostly for its high precision in modeling the performance and power of the Intel XScale core [65]. More precisely, it exhibits an average performance error of 6.5% and an even smaller average power error of 4% [23] compared to real hardware. XTREM allows monitoring of **14 different functional units** of the Intel XScale core: Instruction Decoder (DEC), Branch-Target Buffer (BTB), Fill Buffer (FB), Write Buffer (WB), Pend Buffer (PB), Register File (REG), Instruction Cache (I$), Data Cache (D$), Arithmetic-Logic Unit (ALU), Shift Unit (SHF), Multiplier Accumulator (MAC), Internal Memory Bus (MEM), Memory Manager (MM) and Clock (CLK).

The known topic on whether we should base our profiling study on a SimpleScalar-derivative with all its known bugs and modeling inaccuracies arose also in this case. Except for the obvious issue of availability that we discussed before, the XTREM authors and we, of our own accord, have verified that XTREM has been debugged from SimpleScalar issues and has been largely rewritten to reflect the exact architecture of the XScale processor. Its accuracy has been validated by directly comparing its behavior to that of an actual XScale-based development board. Unfortunately, as it turns out, XTREM suffers from its own bugs and implementation problems – complete errata given at a later chapter of this thesis. Nevertheless, there is no completely bug-free tool out there, especially when non-commercial, research-level tools are considered. Besides, we have tried utilizing – at least partly – a number of SimpleScalar flavors (e.g., Sim-bpred, Sim-fast) while compensating for the simulator errors. Eventually, we chose to abandon them altogether for reasons of modeling accuracy, tool-flow complications and/or simulation accuracy.

[6] Prof. Stefanos Kaxiras, Uppsala University, has referred us to XTREM, developed by Gilberto Contreras, PhD, from the group of Prof. Margaret Martonosi, Princeton University.

The only suitable alternative to XTREM we have come across is called XEEMU [60]. It is, in fact, a largely updated version of XTREM. The authors have fixed a large number of bugs in XTREM, have updated the performance and power models and have made all necessary modifications to better match the XScale pipeline. The reasons we have not replaced XTREM with XEEMU for our experiments are that: (a) XEEMU was not made available until much later than XTREM, and (b) in order to more accurately model the targeted XScale processor, the authors have drastically limited the simulator parameters and their ranges to the ones also encountered in XScale; for instance, no alternative branch-prediction schemes are supported in XEEMU. Apparently, using XEEMU would lead to serious limiting of the SiMS-processor design space under exploration. This conflicts with the first item in our simulator "wish list". Since, at this early point, we are more interested in traversing as broad a design space as possible than achieving maximal simulation accuracy, we have chosen to keep XTREM for our experiments. In the next chapter, the specifics of the simulator will be discussed in more detail.

3.3.3 Evaluation of suitable implant benchmarks

For identifying a suitable benchmark collection for the SiMS processor, a study was, first, required on the best-performing algorithms per category. To this end, prior comparative studies have been investigated. *None* of them has been on the field of biomedical implants. Neighboring fields of resource-constrained systems such as WSNs have been looked into, however implants present distinct traits. To exemplify, although various compression algorithms have been considered, the energy efficiency of data decompression in particular is not our priority in this work since the largest fraction of wirelessly transmitted data in implants is outbound traffic, i.e. telemetry of biomedical data to an ex-vivo monitoring system. Further issues applying to WSNs such as total energy cost for data hopping through a network of nodes do not apply in our case, too.

The particular related work found so far could not be directly transferred to the biomedical-implant field. Nevertheless, in the next sections we chose to summarize the various findings for completeness. We may not have been able to use most of these findings but they have certainly given us good pointers on, for instance, which algorithm aspects to pay attention to during our profiling studies, what metrics to use to evaluate them fairly and so on. In some cases, we have also been able to find algorithm source code which we have directly included in our own studies.

3.3.3.1 Compression algorithms

Barr and Asanovic [11] have worked extensively towards the power trade-off between compression and Tx/Rx power of data on a testbed functionally similar to the popular Compaq iPAQ handheld. Their analysis reveals that with several typical compression algorithms, there actually is a net increase in energy. They propose the use of asymmetric compression, that is, use of a low-energy compression algorithm on the transmit side and a different algorithm for the receive side to cope with the problem.

In the area of WSNs, Maniezzo et al. [86] have worked on surveillance sensor networks and sought to define an online energy trade-off mechanism between compressing image data in a sensor or forwarding (i.e. transmitting) them to the next sensor closer to the base station.

Ferrigno et al. [43] have attempted to balance between local and central data processing in an effort to minimize sensor energy consumption. They have investigated various lossy image-compression algorithms and have made an educated selection based on its performance and energy needs.

Kimura and Latifi [76] have performed a survey on data compression for WSNs and have profiled four compression algorithms specifically designed for WSNs.

3.3.3.2 Encryption algorithms

Much effort has already been spent on the profiling of encryption algorithms in the field of wireless sensor networks (WSNs). Law et al. [79] have evaluated various block ciphers on a MSP430F149 core by Texas Instruments. Their focus has been WSN applications and they have evaluated their included ciphers in terms of security level, operation mode, computational effort and memory requirements. Energy figures have been drawn indirectly from the number of execution cycles needed by each cipher. The authors have proposed best cipher candidates for different combinations of available system memory and desired security level.

Luo et al. [85] have evaluated block and stream ciphers for WSN-nodes in terms of memory requirements and execution time. Chang et al. [20] have attempted energy measurements on RC5, DES and AES running on both the Ember and the CrossBow sensor nodes. Testing various plaintext sizes, they have measured the energy costs of encryption, hashing and wireless transmission of data and assess the reduction in the lifetime of sensor nodes employing

encryption.

Venugopalan et al. [143] evaluate the computational requirements of various stream/block ciphers and hash functions across a wide range of platforms. Based on their findings on the chosen platforms, they attempt to derive a multivariant model which allows the interpolation of performance for other, unevaluated architectures.

Grossschadl et al. [54] have used Sim-Panalyzer [139] to evaluate lightweight versions of RC6, RIJNDAEL, SERPENT, TWOFISH and XTEA in terms of performance, power and memory requirements. Their results indicate that carefully optimized versions of RC6 and RIJNDAEL can preserve their high performance while meeting tight code-size constraints. They have also discussed the impact of key expansion and different modes of operation on the overall performance and energy consumption.

3.3.4 Investigating benchmark suites for implants

After investigating most suitable benchmarks for the SiMS processor, the logical step has been to put a benchmark suite for implant processors together. As will be explained in the next chapter, out of this effort the ImpBench benchmark suite was created. In order to validate the uniqueness and usefulness of this new suite, a large number of existing benchmark suites proposed for various application areas has been investigated.

The SPEC benchmark suite with its latest version, the CPU2006 [128], targets general-purpose computers by providing programs and data divided into separate integer (INT) and floating-point (FP) categories. In particular, the design of server- and desktop-class microprocessors has been heavily influenced by the popular SPEC benchmarks as a measure of performance.

MediaBench [80], now in version II, is oriented towards multimedia- and communications-oriented embedded systems. The authors identify that most advances in compiler technology for instruction-level parallelism (ILP) have focused on general-purpose computing, driven by SPEC-characterized workloads. With the introduction and establishment of a plethora of multimedia-targeted embedded processors provisioned for increased ILP, new workloads needed to be introduced, as well. MediaBench has been put together to address that need.

The Embedded Microprocessor Benchmark Consortium (EEMBC) [1] is a non-profit organization aiming at the development of embedded-systems benchmarks for hardware and software performance evaluation. The con-

sortium licenses "algorithms" and "applications" organized into benchmark suites targeting telecommunications, networking, digital entertainment, Java, automotive/industrial, consumer and office equipment products. It has also provided a suite capable of energy monitoring in the processor. Of late, EEMBC has introduced a collection of benchmarks targeting multicore processors (MultiBench v1.0). However, subject to the consortium licensing regulations, only EEMBC members are entitled to publish their benchmark test results and they can do so by previously submitting these to a certification lab.

MiBench [55] is another proposed collection of benchmarks aimed at the embedded-processor market. It features six distinct categories of benchmarks ranging from automotive and industrial control to consumer devices and telecommunications. According to the authors, MiBench has many similarities with the EEMBC benchmark suite, however it is composed of freely available source code. The diversity and usefulness of MiBench has been evaluated against the SPEC2000 benchmarks.

NetBench [92] has been introduced as a benchmark suite for network processors. It contains programs representing all levels of packet processing; from micro-level (close to the link layer), to IP-level (close to the routing layer) and to application-level programs. The authors show that although they aim architectures similar to ones MediaBench does, their workloads have significantly different characteristics. Hence, a separate benchmark suite for network processors has been considered a necessity.

Network processors are also targeted by CommBench [150], focused on the telecommunications aspect. It contains eight, computationally intensive kernels, four oriented towards packet-header processing and four towards data-stream processing. The suite is evaluated against SPEC95 and its usefulness is shown in a usage case of designing a single-chip, network multiprocessor.

3.3.5 Processor microarchitecture exploration

Having secured a suitable processor-simulation environment and a working set of benchmarks, exploring various aspects of the SiMS processor was made possible. This effort has, nevertheless, been severely limited by the flexibility allowed by the simulator itself. As it were, a couple of micro-architectural features of the processor (cache geometries and branch-prediction schemes) could be sufficiently explored in a systematic way, as will be discussed in Chapter 5. Modifying the processor-simulator ISA has proven to be an unsurmountable task; thus, we have limited our ISA-exploration efforts to a detailed profiling

of the various executed instruction mixes. This has, in fact, been implemented as part of the aforementioned studies on implant benchmarks. Prior art on the explored microarchitecture features is reported next.

3.3.5.1 Evaluation of L1 I-/D-cache organizations

A significant body of prior work has been published on cache behavior with respect to traditional metrics (e.g. cache misses) as well as recent ones (e.g. power or energy).

Fornaciari et al. [46] have proposed a design framework for fast exploration of energy and performance constraints (ED metric) at the system level. Their framework, among others, supports the investigation of I- and D-cache configurations of different cache sizes, block sizes and associativity. Its applicability is limited by the fact that a complete specification of the processor core is needed, which is not available in our case, yet.

Hicks et al. [61] present an exhaustive analysis of power consumption in caches when varying all cache configuration parameters. Unfortunately, their working dataset has been a subset of SPECint92 benchmarks which does not apply in our case of biomedical implants.

Kamble and Ghose [71], on the other hand, have taken a different approach and proposed analytical energy models for caches but their work is not applicable in our case for the same reason as that of Hicks et al..

Givargis et al. [52] have evaluated the power consumption of various cache and bus architectures with parameterizable characteristics.

Su and Despain [136] have performed a case study on power-performance trade-offs for various conventional and new cache designs targeted for low power.

Shiue and Chakrabarti [122] have investigated suitable cache configurations for low-power, embedded systems. They correct and improve on the Kamble-Ghose and Hicks analytical models and propose algorithms for finding optimal configurations.

A problem with the above works is that caches are studied in *isolation* from the rest of the system and, thus, no overall performance behavior is attached to the various power figures, while information about the interplay between different cache configurations and other components of a processor core cannot be acquired. Further, most of the above studies fail to report area as well as energy figures along with the performance results which, as we will see in

Chapter 5, may fail to give the complete picture.

3.3.5.2 Evaluation of branch-prediction schemes

As in the case of cache geometries, a significant body of work has been previously published on branch-prediction behavior with respect to traditional metrics (e.g. accuracy, performance) as well as recent ones (e.g. power, energy, delay).

Skadron et al. [124] have presented an exhaustive analysis of the interaction between branch prediction, instruction-window size and cache size. They have utilized as workloads the SPECint95 benchmarks. Their main focus was in the interplay between the three structures in terms of processor performance.

Youssif et al. [158] have compiled a comprehensive list of currently existing prediction schemes and have evaluated them in terms of performance. Tests have utilized the SPEC2000 benchmarks and a superscalar-processor simulation environment.

Parikh et al. [105] have investigated power repercussions of three advanced branch predictors on a Alpha 21264 simulator under SPEC2000-selected benchmarks. They have, then, proposed three interesting techniques for reducing power consumption in the branch-prediction unit.

Jimenez et al. [69] have approached the branch-prediction issue from the viewpoint of delay as well as, typically, of accuracy and area. To this end, they have proposed three techniques for accommodating delay since they indicate its increasingly dominating impact on performance in future processors with large prediction structures.

All above works fail in either one of two respects: i) they do not study the whole processor when different prediction schemes are utilized and particularly their reaction to different I/D-cache sizes, and ii) they are concerned only with performance as a metric of efficiency. Last but not least, none of these works reports efficient branch-prediction techniques for the particular field of implant processors.

3.3.6 Automated, multiobjective DSE for implant processors

The design space for a processor is huge, and while we would like to cover as much of it as possible, evaluating the space for every single processor configuration possible is virtually impossible. Many general techniques have been

proposed in literature that explore the design space and search for optimal points. Dave [28] provides a concise review of the possible generic optimization techniques and the reasons for selecting a *multi-objective genetic algorithm* (as has been the case in this work). For an more general, in-depth analysis of designing processors from DSE to synthesis, the interested reader can also refer to Gries [53]. In this section, we shall briefly list some tools and techniques developed specifically for DSE of processors.

Hekstra et al. [58] explored the TriMedia CPU64 design using *pruning* – they first probe the design space in order to identify the architectural parameters that affect overall performance the most. The extreme values of these parameters provide "corner cases" and help in bounding the space to be explored in detail.

DESERT [10] (DEsign-Space ExploRation Tool) is a meta-programmable tool for pruning large design spaces using constraints. It represents the design space as a generic structure based on alternatives and parameters and, therefore, can be used for diverse applications. Mohanty et all [95] have used the MILAN (Model-based Integrated simuLAtioN) tool, based on DESERT, for pruning design spaces of heterogeneous multi-core systems. Pimentel et al. [110], who have come up with Artemis (Architectures and Methods for Embedded Media Systems), also work on exploration of heterogenous multi-core environments, but focus more on modeling and simulation than techniques for DSE.

Cho et. al [21] contend that microarchitecture design is better done by considering dynamic behavior of workloads rather than designing for worst-case workload behavior. They use wavelet-based multi-resolution decomposition and neural network based non-linear regression modeling to reason about workload dynamics (in terms of performance, power, and reliability) across the microarchitecture design space.

The PICO [72] framework designed at HP Labs, given C code, outputs application-specific, embedded computer systems optimized for cost vs. performance, where the 'computer system' consists of an EPIC/VLIW and an NPA. It uses a *space walker* – which may be a heuristic, or brute-force algorithm, depending on the search space – to search the design space. A 'component assembler' outputs HDL code for the processors specified by the space walker by assembling low-level components from their *component library*.

Xie et. al [154] provide an in-depth discussion on DSE for 3D-Integrated circuits, including CAD and design tools and simulators. They use *simulated annealing* to automatically find floorplans for the ICs.

Stijn et. al [40] evaluate various automated, single- and multi-objective optimizations for exploring high-performance, embedded, out-of-order processor

designs. They found that a genetic local search algorithm outperforms all other techniques for their application.

Ascia, Catania and Palesi [6] propose using genetic algorithms to perform DSE in processors. They apply a genetic algorithm to optimize the memory hierarchy in terms of area, power and mean access time. However, they use a single-objective genetic algorithm and model the fitness function as a product of the three objectives. Unfortunately, such techniques face drawbacks that reduce their suitability for use with design spaces whose shapes are not known in advance[7], as in our case.

Thiele et. al. [137] present domain-specific DSE for network-processor architectures. They specify models for packet-specific tasks and network traffic ("encoding"), methods to estimate delays and queuing memory ("simulation") and use an *evolutionary algorithm* to perform multi-objective DSE ("optimization").

This last work is perhaps the closest to our work, as it targets domain-specific processors – they focus on networking, we focus on implants – and employs true multi-objective optimizations for the DSE. For this reason, the proposed evolutionary algorithm by Thiele et. al. has indeed been used as the base for designing our own DSE tool, as will be discussed in Chapter 6. However, to the best of our knowledge, DSE with respect to implantable systems has – in the general case – not been previously studied.

3.4 Summary

In this chapter we have primarily introduced the novel SiMS concept. We have outlined, to some extent, the various aspects of the SiMS framework and offered some interesting ideas worth investigating under the SiMS umbrella. We have, then, moved on presenting the organization of the work on the SiMS processor, as performed through the course of this research. We have concluded the chapter with a rather extensive list of related works on each of the topics dealt with in our work. Through the sheer number and diversity of these works, one can readily observe the wide range of topics needed to cover in order to have a first take on the envisioned SiMS processor.

[7] For instance, the design space may well be non-convex, in which case this method does not work.

Note. The content of this chapter is based on the the following papers:

C. Strydis, G. N. Gaydadjiev, S. Vassiliadis, **A New Digital Architecture for Reliable, Ultra-Low-Power Systems**, ProRISC 2006, pp. 350-355, Veldhoven, The Netherlands, November 2006.

C. Strydis, G. N. Gaydadjiev, S. Vassiliadis, **A Generic Digital Architecture & Compiler for Implantable Devices**, Architectures and Compilers for Embedded Systems (ACES 2005), Ter Elst, Edegem, Belgium, September 2005.

4

SiMS-processor simulation environment

BEFORE any processor specification and design phase can commence, a proper simulation and evaluation environment needs to be established, first. The environment may consist of one or more suitable simulators which need to be fed realistic workloads to execute, if any meaningful results are to be obtained. The effort is considered successful if the profiling results lead to the design of a processor which – when fed the same workloads as the simulator(s) – will produce identical or similar results. This, however, is a typical *chicken-and-egg problem*: we wish to design a new processor and require an accurate simulation environment to do so, however, we cannot *a priori* know the accuracy of the environment before we actually build the envisioned processor, and so on.

Selecting (and designing, for that matter) accurate simulation tools is, on its own, a considerable problem. Experience in a particular design field is invaluable for making better selections. Unfortunately, in the field of biomedical implants – as seen from the standpoint of SiMS – there is no such experience or prior art available or, at least, documented. The only insights to be drawn upon have come from the implant survey we have performed in Chapter 2 and from our standing experience on the broader fields of computer architectures and embedded systems. As will be revealed in this and following chapters, setting up a consistent simulation/evaluation environment has taken up a considerable amount of our time which is comparable to the actual exploration process of the SiMS processor. We have consciously made this choice so as to lay a solid groundwork upon which our own, limited by necessity, as well as other, more extended, future SiMS-processor, exploration efforts can be based.

In line with these considerations, this chapter is occupied with defining the *simulation environment* – simulator, benchmarks, input datasets – for our further experiments. The simulator employed (as introduced in the previous chapter)

is detailed and practical issues are discussed. Since no reference work exists for an established benchmark base, original benchmark programs (able to run in the simulator) are being investigated and the most suitable ones (in terms of multiple metrics) are grouped in a *novel benchmark suite* for implant processors, called ImpBench. Proper input datasets to these benchmarks are also discussed. With all pieces of the simulation environment finally in place, we conclude the chapter with the *case study* of an instance SiMS-processor application.

4.1 XTREM processor simulator

4.1.1 Hardware-modeling details

As mentioned in the previous chapter, for our experiments we have used XTREM [24], a modified version of SimpleScalar/ARM [8, 18]. The XTREM simulator is a **cycle-accurate**, **microarchitectural**, **performance-** and **power-simulator** for the high-performance, low-power Intel XScale core [65]. It models the effective switching-node capacitance of various functional units inside the core, following a similar modeling methodology to the one found in Wattch [17]. XTREM boasts high precision in performance and power modeling; more precisely, it exhibits an average performance error of 6.5% and an even smaller average power error of 4% [23] compared to real hardware.

The main XScale (and XTREM) characteristics are summarized in Table 4.1. Many of its (micro)architectural features have been integrated into XTREM. Thumb instructions and special memory-page attributes are not supported but they do not affect simulation results since they are not used by our benchmarked applications. XTREM allows monitoring of **14 different functional units** of the Intel XScale core: Instruction Decoder (DEC), Branch-Target Buffer (BTB), Fill Buffer (FB), Write Buffer (WB), Pend Buffer (PB), Register File (REG), Instruction Cache (I$), Data Cache (D$), Arithmetic-Logic Unit (ALU), Shift Unit (SHF), Multiplier Accumulator (MAC), Internal Memory Bus (MEM), Memory Manager (MM) and Clock (CLK).

Since XScale (and, thus, XTREM) is a low-power processor with aggressive power-management features, we are well-aware that it is not perfectly suitable for biomedical implants in terms of power consumption. However, our selection has been based on *availability* and on the crucial fact that XTREM *models actual hardware* with very high accuracy using hardware performance counters (HPCs). Particularly for the cache(-like) structures incorporated in

feature	value
ISA	32-bit ARMv5TE-compatibility, 8 DSP instructions
Pipeline depth	7/8-stage (depending on instruction), super-pipelined
Datapath width	32-bit
RF size	16 registers
Issue policy	in-order
Instruction window	single-instruction
I-Cache	32KB 32-way set-associative (1-cc hit/170-cc miss lat.)
D-Cache	32KB 32-way set-associative (1-cc hit/170-cc miss lat.)
TLB	32-entry fully-associative
BTB	128-entry direct-mapped
Branch Prediction	2-bit Bimodal
Write Buffer	8-entry
Fill Buffer	8-entry
Memory bus width	4-byte
INT/FP ALUs	4/4
DSP co-processor	40-bit, low-power, variable-lat. MAC
Clock frequency	2 MHz (typically 200 MHz)
Operating voltage	1.5 Volt
Implementation technology	0.18 μm
Sampling period	10,000 cc's (typically 200,000 cc's)

Table 4.1: *XScale architecture details.*

the simulator, analytic power models have been developed and their accuracy has been verified [23]. In order to match our application field better, we have – through the process of our research – limited or disabled many of XTREM's architectural parameters. As can be seen in Table 4.1, clock frequency has readily been reduced from 200 *MHz* which is the preset frequency in XTREM, to 2 *MHz* to closer resemble realistic implantable systems. Throughout our analysis, we have iteratively tuned the XTREM parameters to optimal settings, subject to our ongoing findings. We will take the time to mention the exact settings used in every phase of our exploration study. Even with such adjustments, power and performance results should *not* be taken as absolute figures but as relative measures of processor behavior across different input datasets, workloads and microarchitectural configurations.

4.1.2 Program-execution details

Since XScale supports the **ARM ISA** and XTREM has, subsequently, been based on a modified version of SimpleScalar/ARM, XTREM (ELF) binaries should be built with a version of the GNU ARM-GCC (or any other ARM cross-compiler, for that matter). We found out that a *tweaked* version of the ARM cross-compiler is needed as XTREM does *not* accept standard ARM

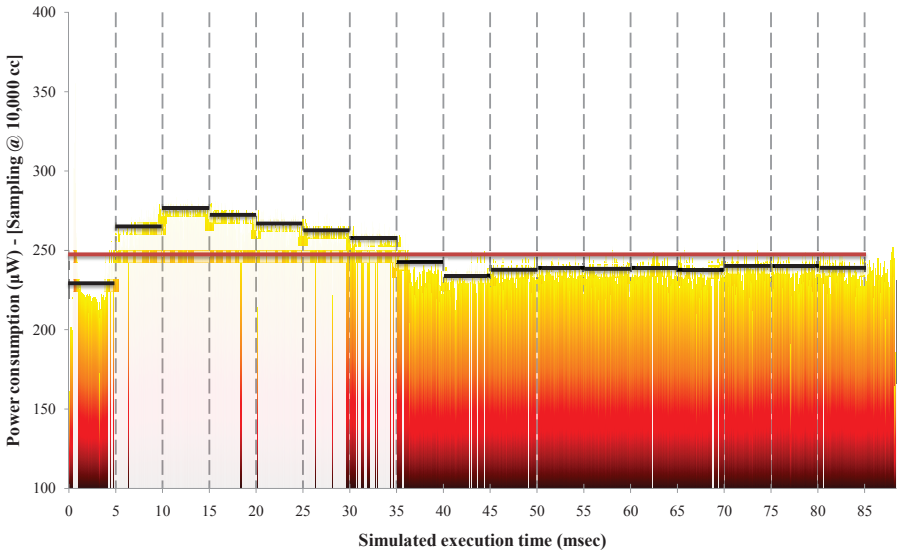


Figure 4.1: *XTREM-generated power profile for single program execution.*

binaries. Setting up a cross-compiler for this purpose has proven somewhat of a challenge and, eventually, a modified version of GNU ARM-GCC v2.95.2 has been used, as provided in the SimpleScalar/ARM webpage[1]. Through this setup, a working (if old) C compiler and a full complement of binary utilities (assembler, disassembler, linker, object dump etc.) has been established.

4.1.3 Sampling details

XTREM bases its performance- and power-modeling capabilities on HPCs. Over a certain sampling period, HPCs are being updated and when this period has expired, the counters' contents are fed into the performance and power models and are, then, reset for the next period. This period has been hard-coded in XTREM and initially set at 200,000 clock cycles (cc's), as can be seen in Table 4.1. Early in our experiments – and at the cost of significantly increased simulation times – we have lowered that period to 10,000 cc's in order to achieve more accurate performance and power profiles over program execution time. To illustrate, XTREM provides power (and, in a similar fashion, performance) results as the ones shown in Figure 4.1. Each vertical, black bar represents an XTREM sample which is power consumption averaged over

[1] Available online: `http://www.simplescalar.com/v4test.html`

10,000 cc's.

As we shall see in the rest of the chapter, the necessary dimensions of our experiments – multiple benchmark runs, multiple input datasets – and the sheer volume of sampled data forces us to aggregate hundreds of thousands of these samples into a *single*, *average* value (red line in Figure 4.1) characterizing the execution of a single benchmark. Then, the following question arises: How do we calculate this *aggregate average power consumption* from the XTREM-generated power samples?

Let us assume $f(t)$ is a function limited in the interval $[a, b]$ and generally continuous or monotone.

Property 4.1

$$\int_a^b f(t)dt = \int_a^c f(t)dt + \int_c^b f(t)dt$$

with a, b, c any internal points of the interval of integration.

Property 4.2

$$\int_a^b kf(t)dt = k\int_a^b f(t)dt$$

if k is a constant $\neq 0$.

Property 4.3

$$\int_a^b [f_1(t) + f_2(t) + \ldots + f_n(t)]dt = \int_a^b f_1(t)dt + \ldots + \int_a^b f_n(t)dt$$

if $f_1(t), f_2(t), \ldots, f_n(t)$ are integrable functions. A constant function is an integrable function.

Let us consider the interval $[0, E_x]$ and let $0 = t_0 < t_1 < \ldots < t_n = E_x$ be a partitioning of $[0, E_x]$ with $t_{i+1} - t_i = \alpha \in \mathbb{R}^+ \; \forall i$. Let $P(t)$ be the function defined as follows:

$$P(t) = \begin{cases} k_1 & \forall t \in [t_0, t_1] \\ k_2 & \forall t \in [t_1, t_2] \\ \ldots & \ldots \\ k_n & \forall t \in [t_{n-1}, t_n] \end{cases} \tag{4.1}$$

We have the following:

Theorem 4.4

$$\int_0^{E_x} P(t)\,dt = E_x \cdot P_{Average}\,. \tag{4.2}$$

Proof.

$$\int_0^{E_x} P(t)\,dt =_* \int_{t_0=0}^{t_1} k_1(t)\,dt + \ldots + \int_{t_{n-1}}^{t_n=E_x} k_n(t)\,dt =$$

(* = Properties (4.1) and (4.3). But $k_i(t)$ is constant on its interval of definition.)

Therefore:

$$= k_1 \int_{t_0=0}^{t_1} dt + \ldots + k_n \int_{t_{n-1}}^{t_n=E_x} dt =$$

We have $t_{i+1} - t_i = \alpha \in \mathbb{R}^+ \; \forall i$, therefore:

$$= k_1\alpha + \ldots + k_n\alpha = \alpha \sum_{i=1}^{n} k_i\,.$$

We have:

$$E_x = n\alpha \Rightarrow \alpha = \frac{E_x}{n} \tag{4.3}$$

Moreover, we have:

$$P_{Average} = \frac{\sum_{i=1}^{n} k_i}{n} \tag{4.4}$$

Therefore, by combining equations (4.3) and (4.4), we finally have:

$$\int_0^{E_x} P(t)\,dt = \alpha \sum_{i=1}^{n} k_i = \frac{E_x}{n} \sum_{i=1}^{n} k_i = E_x P_{Average}\,,$$

with $P_{Average}$ the *arithmetic mean* of the XTREM-generated power samples. ■

4.2 Implant workloads

Although, in the widest sense, biomedical implants are embedded systems, they adhere to a unique set of design and operation requirements which dictate their own design space and workloads. Some of the most prominent implant-workload characteristics are discussed next.

4.2.1 Workload characteristics

First off, as illustrated in Section 2.6.1, a large class of biomedical implants performs *periodic*, in-vivo measurements of physiological data (blood pressure, blood temperature, intracranial pressure, blood-glucose concentration, muscle or nerve activity etc.) through appropriate sensors. The collected data need either be *stored* inside the implant for later telemetry to an external monitoring device, e.g. a treating physician's office computer, or to be periodically *transmitted* to an external data-logging system such as a PDA, laptop computer etc.. This pattern of behavior indicates that outbound biological-data traffic almost always dominates inbound traffic. Besides, data must be transmitted *securely* and *reliably*, meaning that information eavesdropping or loss is not tolerated.

Secondly, depending on the application, implant processors may need to perform computation-, control- or I/O-intensive tasks in the human body, for instance, *collection* of sensory readouts, *processing*, *storage* and open- or closed-loop *control* of bio-actuators. In all cases, throughput should be no higher than that required by the underlying application for maintaining a low as possible energy profile and a highly reliable operation. Autonomous operation and dependability are primary concerns in implantable systems given the health and economical implications at stake.

Thirdly, biological or other data manipulation in implants can in most cases be coped with through *integer arithmetic*. Expensive, *floating-point* operations can be avoided by smart manipulation of the data or postponed until such time when data are telemetered to an external logging station with infinite (in our context) computational resources, thus saving the implant the trouble of processing them. There are, however, distinct cases where in-vivo, run-time decisions have to be made depending on the results of floating-point math operations.

4.2.2 Identifying generic workloads

On proposing a new, generic approach to implantable systems – and their PCCs, in particular –, it becomes necessary to define a representative number of workloads that such PCCs will be executing. As with most other aspects of the SiMS project, we had to start from scratch and build this workload (or benchmark) *suite* based on hints from the surveyed implantable systems and our own intuitions on the matter.

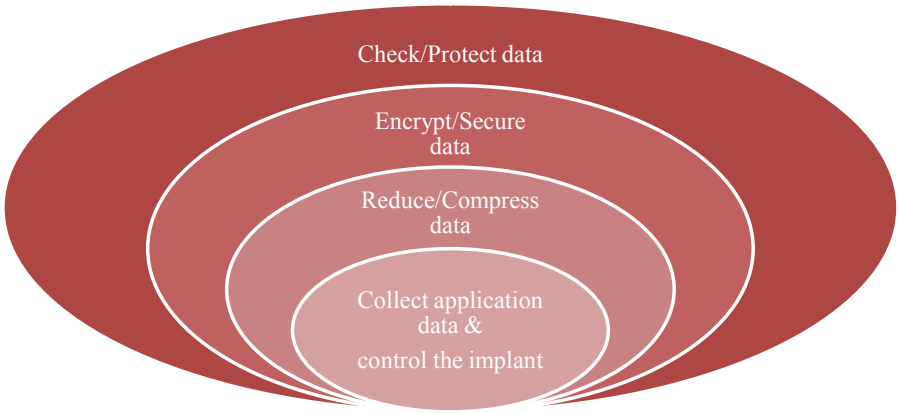


Figure 4.2: *Envisioned standard tasks of implant processors.*

4.2.2.1 Real implant applications

In Figure 4.2, we illustrate the general composition of implant workloads, as anticipated to be in the coming years. In the center of these workloads lies the per-instance, *real implant application* comprising some sensing/actuating functionality in open- or closed-loop fashion. This functionality generally consists of:

a) reading in-vivo physiological data from the body and/or commands from an external host;

b) assessing their meaning, processing them and/or responding with suitable actions such as stimulation of muscles or nerves, release of drug in the blood stream, and so on;

c) storing collected data locally (i.e. data-logging) and/or transmitting them to external monitoring stations; and

d) properly coordinating the implant peripherals to achieve all the above.

The precise functionality varies largely with the medical condition being treated and is analogous to the wide diversity of applications targeted by implantable systems. We, thus, have systems performing any part or the whole of the above tasks. We have to support all the above tasks to a satisfactory degree in order to design generic SiMS systems. Although difficult to capture in full detail, we have attempted to collect implant applications that encompass the full range of activities (a) to (d).

4.2.2.2 Data reduction & compression

As the survey has revealed (see Section 2.6.4), the number of attached peripherals, the resolution (in bits) of employed ADC units as well as the amount of processing performed in implants appears to be increasing steadily over time. However, the highly constrained nature of implantable systems poses serious size and power limitations on the amount of data that can be stored inside the implant or wirelessly transmitted to an external host.

Based on this observation, we are inclined to consider data compression and/or reduction as a significant task of future implants, as shown in Figure 4.2, even though only a couple of surveyed systems with such workloads have been encountered (Wang L. et al. [145], Eggers et al. [37]). While data reduction can probably be performed locally – at the point of data acquisition (i.e. at the sensor output) at very low cost –, efficient data compression (and expansion) would require large stored datasets to operate on and more extended processing. What is more, data reduction is highly application-dependent and general techniques cannot be selected for use as typical workloads. For these reasons, we have chosen to focus more on compression algorithms.

4.2.2.3 Data & command encryption

The personal, sensitive nature of collected (and transmitted) physiological data calls for some way of securing them against any unauthorized individual. Eavesdropping on the data may not only cause personal and social problems to the implant user but, more crucially, may allow loss of implant control to a malevolent interloper. Accordingly, another task we think will become a necessary and, thus, frequently executed workload in future implants is information encryption, as illustrated in Figure 4.2.

Up till now, no existing implantable system has been provisioned with encryption capabilities and we consider our work the first to study encryption algorithms for this purpose. Although this might sound as utter science fiction, indications from more mature application fields, such as mobile telephony, teach otherwise: It is a simple matter of time until implants become so advanced and wide-spread that they will be plagued with frequent security attacks.

4.2.2.4 Data & command integrity

Incoming command frames to the implant and outgoing data frames to external hosts are usually taking place in a rather hostile environment. Wireless transmissions have to be robust enough to withstand command/data sending through layers of bodily fluids as well as soft and hard tissue and, finally, air and any other natural obstacles. All this is occurring in the presence of various EMI sources and under low information-signal SNR-levels, mainly due to low available power budgets. These unfavorable conditions require some sort of command- and data-frame protection mechanisms operating on the frames after encryption has taken place. This is illustrated in Figure 4.2. Most prominent information-redundancy mechanisms are techniques such as CRC- and checksum-calculating algorithms.

Contrary to the previously discussed workloads, maintaining information integrity has occupied various surveyed implantable systems. Namely, Lanmüller et al. [77, 78], Wang W.X. et al. [146], Fernald et al. [41] and D'Lima et al. [32] have been found to employ CRC schemes for robust, error-free implant communications. Such schemes have been shown in, at least, two cases (Lerch et al. [83], Stangel et al. [130]) to have been implemented as internal-processing tasks of implants. Since such schemes are so important for maintaining information integrity, especially in a field where information loss is almost always unacceptable, we have included suitable workloads as the final step after data-collection, compression and encryption tasks.

4.2.3 Workload acquisition

Since we selected XTREM, a cycle-accurate simulator which executes real ARM code written (and compiled) in C, the selected implant workloads have to be encoded as appropriate programs in order to be fed to it. These programs must be written in C (ANSI C, actually) and must be driven by realistic input datasets.

This task has proven more difficult than initially perceived. The reason is that – as real implant functionality goes – the survey performed in Chapter 2 was mostly unable to tap into C source code (or any other language, for that matter) that could be used for feeding XTREM. Executed code was either not reported or, simply, not applicable; for instance, naive assembly code of limited usefulness (triggering various external or internal interrupts) was provided which cannot be accurately modeled in a processor simulator such as XTREM. As far as compression and encryption algorithms are concerned, since we are the

dataset type	BSL test	dataset name	BIN size (B)	ASCII size (B)	samples (#)	duration (sec)	samp. rate (sml/sec)	samp. rate (KB/sec)
Electromyogram II (EMG)	John-L02	**EMGII_01**	1152	1778	144	0,288	500	3,89
Electromyogram II (EMG)	John-L02	**EMGII_10**	9608	14095	1201	2,402	500	3,91
Electroencephalogram I (EEG)	John-L03	**EEGI_01**	984	1332	123	0,615	200	1,56
Electroencephalogram I (EEG)	John-L03	**EEGI_10**	9616	13005	1202	6,01	200	1,56
Electrocardiogram I (ECG)	John-L05	**ECGI_01**	912	1406	114	0,114	1000	7,81
Electrocardiogram I (ECG)	John-L05	**ECGI_10**	9616	14824	1202	1,202	1000	7,81
Respiratory Cycle I (RC)	John-L08	**RCI_01**	1192	1770	149	1,49	100	0,78
Respiratory Cycle I (RC)	John-L08	**RCI_10**	9520	14581	1191	11,91	100	0,78
Pulmonary Function I (PF)	John-L12	**PFI_01**	1184	1617	148	1,48	100	0,78
Pulmonary Function I (PF)	John-L12	**PFI_10**	9240	12863	1155	11,55	100	0,78
Skin Temperature (AEP)	John-L15	**AEP_01**	1120	1397	140	0,7	200	1,56
Skin Temperature (AEP)	John-L15	**AEP_10**	9736	11739	1217	6,085	200	1,56
Blood Pressure (BP)	John-L16	**BP_01**	1128	1404	141	0,282	500	3,91
Blood Pressure (BP)	John-L16	**BP_10**	9584	12798	1198	2,396	500	3,89

Table 4.2: *1-KB and 10-KB physiological datasets. Double-precision (8-Byte) data samples are used.*

first to consider them as standard implant workloads, we also had no reference codes to use and had to write our own or modify ones found from other sources. Last, information-integrity algorithms considered (CRC and simple checksum) have rather straightforward implementations and were also borrowed from other sources. In the next section, we discuss our selection of realistic input datasets and, afterwards, we delve into the evaluation and selection of suitable implant workloads and we present ImpBench, the first benchmark suite for characterizing generic, biomedical-implant processors.

4.3 Input datasets

Typical biological signals are often highly periodic signals (e.g. heart beat) or stable signals (e.g. blood temperature) which can, under specific circumstances, display gradual or abrupt changes in value (e.g. a sudden muscle contortion). We have collected and used various representative *input datasets* capturing both stable as well as rapidly changing patterns.

Existing literature in the field does not offer any solid pointers or even hints towards a ”standard”, representative dataset. Of course, a lot of biological-signal databases exist and are even available online. However, since there are no solid hints for favoring one dataset source over another, the datasets selected have been extracted from the BIOPAC (R) Student Lab PRO v3.7 (BSL) software for reasons of availability and simplicity. They have the technical specifica-

tions shown in Table 4.2. Our performed implant survey has revealed typical data-memory sizes inside the implants to be less than but approaching $1\ KB$ in size, for both measurement or stimulation applications. However, as Figure 2.26 has revealed, memories are increasing at a steady pace over time. To accommodate for current as well as future implant designs, we have chosen to use datasets of $1-KB$ as well as of $10-KB$ size throughout our experiments.

From the previous table, we can see that typical signals of muscle activity (EMG), heart activity (ECG) and brain neural activity (EEG) as well as breath oxygen volume (RC), lung volume (PF), skin surface temperature (AEP) and blood pressure (BP) have been collected. All considered datasets are representing one-dimensional physiological-signal measurements of varied sampling rates, depending on the particular application. These datasets are actually double-precision, floating-point numbers and have been stored in both ASCII- and binary-encoded files:

- 8-bit ASCII representation: Each readout is a text string terminated (on each line) with CR (carriage-return) and LF (line-feed) characters. Each string consists of as many Bytes as the number of digits plus two extra Bytes for the decimal point and the sign characters; thus, about 8 to 10 Bytes are used per entry, in the general case.
- Binary representation: Each readout is a packed, double-precision, FP number (IEEE-754) of 64 bits; thus precisely 8 Bytes are used per entry.

Most of the input datasets come with a sufficiently extensive description of the experimental setup and the acquisition parameters, as shown in Table 4.3. In the plots of Figure 4.3, the various datasets for 1-KB data are plotted (amplitude vs. time) along with their first-order derivatives, to give a feeling of the morphology of the various datasets as well as to illustrate the rate of change of their values:

From these plots and the preceding tables, it becomes apparent that all datasets are distinctly different in terms of morphology; i.e. amplitude, duration and rate of change, as well as sampling rate. This has been our original intention in defining a diverse yet self-contained set of input vectors and, subsequently, application scenarios.

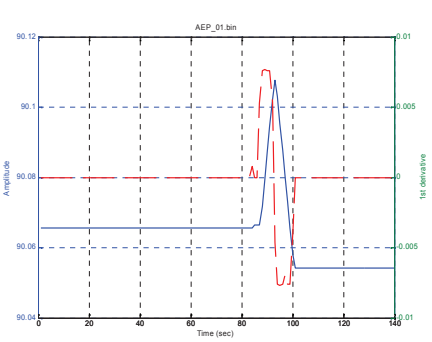
dataset type	description
Electromyogram II (EMG) - 1KB	First procedure, Forearm 1 (dominant): Increased clenches in increments of 5 Kg until maximum clench force is obtained.
Electromyogram II (EMG) - 10KB	Second procedure, Forearm 1 (dominant): Continued maximal clench until fatigue causes 50% reduction in measured force.
Electroencephalogram I (EEG) - 1KB	This recording shows the Subject in a relaxed state, with eyes closed for about 10 seconds. The eyes were then opened for approx. 12 seconds, closed approx. 12 seconds, then opened for the remainder of the recording time.
Electroencephalogram I (EEG) - 10KB	N/A
Electrocardiogram I (ECG) - 1KB	First procedure: Relaxed, lying down. Second procedure: Relaxed, sitting up. Third procedure: Relaxed, sitting up state, taking prolonged breaths. Markers at the start of inhales and exhales. Fourth procedure: Relaxed, sitting up state, recovering from exercise.
Electrocardiogram I (ECG) - 10KB	N/A
Respiratory Cycle I (RC) - 1KB	First procedure: Seated in a chair and breathing normally. Second procedure: hyperventilation for 30 seconds, then recovery. Third procedure: hypoventilation, then recovery. Fourth procedure: coughing, then reading aloud.
Respiratory Cycle I (RC) - 10KB	N/A
Pulmonary Function I (PF) - 1KB	This recording shows normal breathing for 3 breaths, full inhale, return to normal breathing, full exhale, then a return to normal breathing. Note that a Residual Volume of 1.0 Liters was used.
Pulmonary Function I (PF) - 10KB	N/A
Skin Temperature (AEP) - 1KB	Aerobic Exercise Physiology
Skin Temperature (AEP) - 10KB	N/A
Blood Pressure (BP) - 1KB	First procedure: Subject has cuff on LEFT arm, and is sitting up at rest. Second procedure: Repeat of the first procedure. Third procedure: Subject has pressure cuff on RIGHT arm, and is sitting up at rest. Fourth procedure: Repeat of the third procedure. Fifth procedure: Subject is lying down at rest, with pressure cuff on RIGHT arm. Sixth procedure: Repeat of the fifth procedure. Seventh procedure: After mild exercise, Subject should sit up to recover with the pressure cuff on the RIGHT arm.
Blood Pressure (BP) - 10KB	N/A

Table 4.3: *Experimental setup and acquisition parameters of input datasets.*

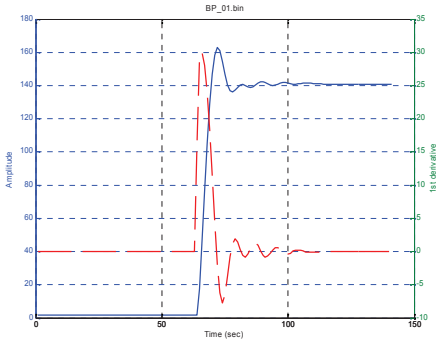
4.4 Profiling of encryption algorithms

As previously discussed, encryption is expected to become a typical implant workload in the coming years. In the current section, we profile – through detailed simulations – a large set of popular encryption algorithms (also known as **ciphers**) in terms of power consumption, energy expenditure, encryption rate and program-code size. We, then, select the ones with the best characteristics for the implant domain and investigate their respective instruction mixes in order to gain insight on the most suitable instructions for inclusion in our targeted SiMS-processor architecture.

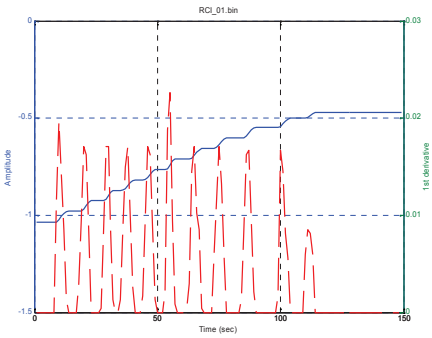
In this profiling study, we are attempting to present a detailed comparison of various encryption algorithms in terms of performance, power etc.. As a second step, we analyze the instruction mix and frequency of the best ciphers and draw directions for the architectural design of the SiMS processor.



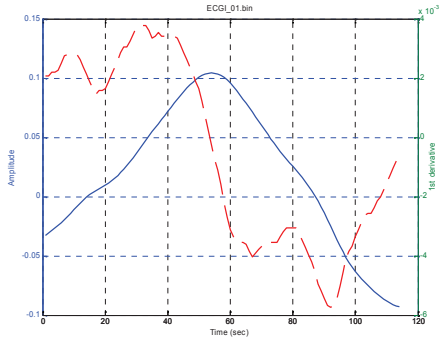
(a) AEP_01



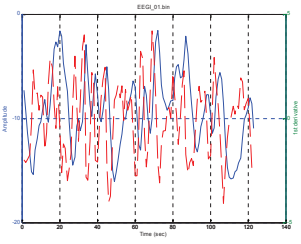
(b) BP_01



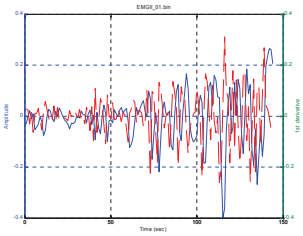
(c) RCI_01



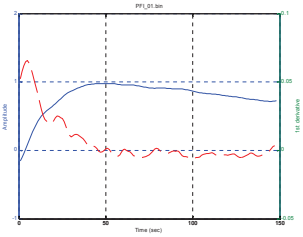
(d) ECGI_01



(e) EEGI_01



(f) EMGII_01



(g) PFI_01

Figure 4.3: *Dataset amplitude vs. time (blue, solid line) and first-order derivative of amplitude vs. time (red, dashed line).*

4.4.1 Selection criteria of ciphers

We have chosen to profile only *symmetric-encryption algorithms* for two main reasons. First, asymmetric schemes have been extensively investigated in the past and, due to their complexity, have been found to have computational and memory requirements that are prohibitively high for low-power, embedded devices [33, 68]. Lately, there has been considerable work in the field, especially in WSNs, showing that carefully optimized software or hardware implementations of existing asymmetric algorithms may be viable for resource-constrained devices [47, 144].

Second, our choice has also been based on communication patterns of typical, *battery-powered* implant applications: Data and, especially, command exchange with the implant does not happen particularly often (e.g. a few times a day or less). This low interaction is desired for reasons of autonomous, *unattended operation* as well as for reasons of *prohibitive energy costs* incurred when wirelessly transmitting information in-vivo.

Asymmetric algorithms are slower and generally less secure than symmetric ones, yet they have the benefit of not sharing (thus, not jeopardizing) a common secret key. Even if a combination of asymmetric and symmetric-key encryption is assumed for achieving more secure authentication and data exchange, respectively, authentication is not expected to occur so often during normal implant operation. It is, thus, not our primary concern for this profiling study which is focused on the most commonly executed task, i.e. the symmetrically-encrypted data exchange.

Another characteristic of typical implant applications is that (outbound) data telemetry takes place a lot more often than (inbound) command reception in implants. In effect, we are focused here only on the *encryption part* of the profiled algorithms. Furthermore, due to their symmetric nature, most of these algorithms have the same computational requirements for both encryption and decryption.

Operation mode is the way for encrypting a message longer than the block size of an algorithm. In this work we only consider the Electronic CodeBook (ECB) mode. It has been shown that different operation modes (e.g. CBC, CFB, OFB) incur different fault-tolerance levels with regard to information loss due to transmitted-packet loss but incur *the same energy penalty* [79]. Since in this work we are not investigating the efficiency of different modes in terms of information integrity but, rather, profile ciphers based on their power signatures, ECB is sufficient.

feature	value
ISA	32-bit ARMv5TE-compatibility, 8 DSP instructions
Pipeline depth	7/8-stage (depending on instruction), super-pipelined
Datapath width	32-bit
RF size	16 registers
Issue policy	in-order
Instr. window	single-instruction
I-Cache	32KB 32-way set-assoc. (1-cc hit/170-cc miss lat.)
D-Cache	32KB 32-way set-assoc. (1-cc hit/170-cc miss lat.)
TLB	32-entry fully-assoc.
BTB	128-entry direct-mapped
Branch Pred.	2-bit Bimodal
Write Buffer	8-entry
Fill Buffer	8-entry
Mem. bus width	4-byte
INT/FP ALUs	4/4
DSP co-proc.	40-bit, low-power, variable-lat. MAC
Clock freq.	2 MHz (typ. 200 MHz)
Oper. voltage	1.5 Volt
Implem. tech.	0.18 μm

Table 4.4: *XTREM configuration for the encryption profiling study.*

4.4.2 Experimental setup

4.4.2.1 Simulator configuration

The main XTREM characteristics, as configured for the encryption profiling study, are summarized in Table 4.4. As the table reveals, clock frequency has been lowered to better resemble realistic implantable systems. Yet, other parameters have not been tampered with since, at the point in time of this first profiling study, it had not been certain whether the simulator will scale properly in terms of performance and power. For instance, instruction and data TLBs have not been disabled, the operating voltage or the memory latencies have not been altered.

4.4.2.2 Encryption datasets

The nature of the input datasets (also known in the encryption field as **plaintexts**) does not affect the behavior of the studied encryption algorithms except for their size. For other algorithms, such as data compression, the dataset nature does impact performance, as will be discussed later, in a compression profiling study. However, for completeness we have evaluated the encryp-

encryption algorithm	block size (bits)	key size (bits)	Rounds (#)
3WAY [2]	96	96	11
BLOWFISH [2]	128	128	16
DES [2]	64	56	16
GOST [2]	64	256	32
IDEA [2]	64	128	8.5
LOKI91 [125]	64	64	16
RC5 [2]	64	128	12
SKIPJACK [125]	64	80	32
XXTEA [148]	64	128	32
MISTY1 [79]	64	128	8
RC6 [79]	128	128	20
TWOFISH [79]	128	128	16
RIJNDAEL [79]	128	128	12

Table 4.5: *Collection of profiled symmetric ciphers.*

tion algorithms using the two BP biological datasets (sizes `1` *KB* and `10` *KB*, roughly) from Table 4.2, representing continuous blood-pressure readouts.

4.4.2.3 Encryption algorithms

When putting together our collection of ciphers, we have made an effort to include sources adhering to the following characteristics:

i. large range of symmetric-encryption techniques and styles, from high-performing to compact flavors;

ii. mature, optimized, well-documented implementation code base;

iii. various algorithmic complexities;

iv. suitability: the XTREM simulator can only handle C and Java sources. Furthermore, in its current version it does not support simulating an OS on top of the simulated hardware, thus prohibiting the use of encryption sources - such as the excellent bzip2 algorithm - that require multithreading support or other high-level features; and

v. availability: all collected ciphers comprise utterly free, published or free under the GNU General Public License sources, readily available to the research community.

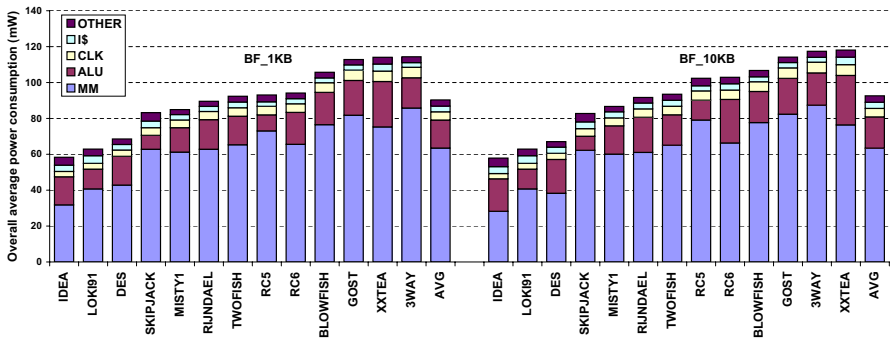


Figure 4.4: *Per-component, average power consumption (in mW) for two plaintext sizes.*

The implementation of a given cipher plays as crucial a role for the performance and behavior of the cipher as its underlying structure. While adhering to the above characteristics, in order to offer the best possible fairness in our selection process, we have attempted to include algorithms built with the same implementation philosophy (e.g. algorithm suite implemented by the same author(s)) and/or algorithms being top representatives in their category. Table 4.5 summarizes our selected cipher collection[2]. The DES algorithm, although not considered secure any longer, has been included in our study as a reference algorithm for the rest of the considered ciphers.

4.4.3 Profiling analysis

4.4.3.1 Power consumption

We start our profiling study by, first, examining how the selected ciphers perform in terms of power consumption since this is a crucial attribute of energy-constrained devices like implants. Overall and per-component average power consumption is depicted in Figure 4.4 for all 13 ciphers and for the two BP plaintext sizes 1 KB and 10 KB.

We can readily see in the figure that, across all ciphers, the memory-manager unit (MM) is the most power-hungry component of the processor with a rough 69% fraction of overall power consumed. The MM unit is activated each time the core is stalled because of a main-memory instruction or data access. Through the course of our profiling studies, we have found out that XTREM

[2]Available online, on the SiMS website: `http://sims.et.tudelft.nl`

uses plain lookup tables with fixed power-consumption figures for some of its subsystems that were difficult to model analytically. One such subsystem is the MM which does not scale properly with operating frequency. This leads to the dominating MM power profiles displayed in this and following sections. We have tolerated this problem throughout since we are interested in the relative power profiles of the various investigated algorithms. However, this means that the high exhibited percentage for the MM power consumption is, in fact, lower.

Next, follow the ALU consuming roughly 18%, the clock structure (CLK) consuming 5% and the instruction-cache (I$) consuming 3.5% of the overall power, on average. Compared with other types of workloads, e.g. data compression, encryption is more computationally intensive (i.e. many arithmetic and logic operations), thus the high consumption of the ALU is not surprising. Furthermore, encryption is typically data- rather than control-intensive, with few instruction branches, placing high demands on linear instruction fetch. That is why the instruction-cache consumes on average more power than other memory units, e.g. the data-cache or the BTB. Last, the clock structure is known throughout digital systems to be a significant component of power consumption, which is also the case here. If we operated the processor at a higher frequency, power consumption would increase considerably. In terms of plaintext sizes, overall average power consumption increases insignificantly (about 3%) with input size. Essentially, in the range from 1KB to 10KB of plaintext size which is of interest for our case, power consumption does not seem to be affected. This agrees also with the findings of Law et al. [79]. In accordance to the same work as well as our own measurements, a significant difference in consumed power would be observed in plaintext sizes comparable to the block size of the ciphers, i.e. 10 to 30 Bytes. In this range, key-initialization tasks place a computational overhead comparable to the actual encryption process. This indicates that encryption becomes more power-efficient with larger plaintext sizes.

A final observation from Figure 4.4 is that the power-behavior of the ciphers does not change with increasing plaintext size, at least in the range of interest. There is one exception: 3-WAY and XXTEA switch places when moving to the larger plaintext but this is of minimal significance since they both score the poorest in terms of average power consumption. The best performing ciphers on this metric are IDEA, LOKI91, SKIPJACK, MISTY1 and RIJNDAEL. Although DES is included in the study as a reference algorithm, it cannot be selected as a winning candidate in the profiling due to its compromised status. It is interesting, however, to observe that it features one of the lowest power

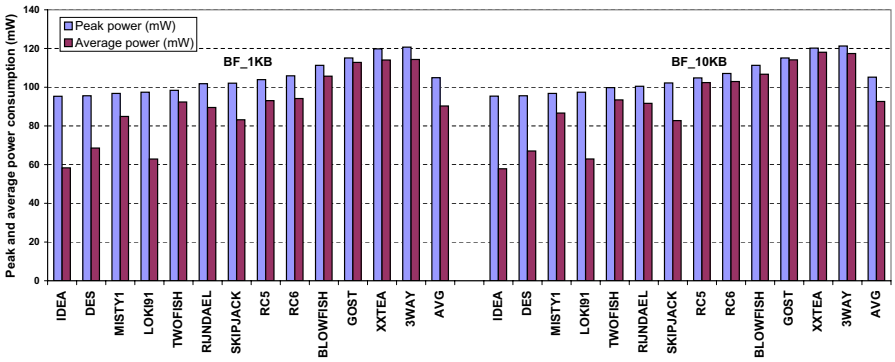


Figure 4.5: *Average and peak power consumption (in mW) for two plaintext sizes.*

profiles even though it is one of the oldest encryption algorithms.

Except for average power consumption, another interesting metric is peak power consumption. This is especially important for battery-powered systems such as implants. A battery able to support a cipher with a given average power consumption may be unable to deliver the required output at a given point in time if the cipher sporadically presents peak power values which are largely deviating from its average power needs. To address this aspect of the profiled ciphers, we have plotted Figure 4.5. The ciphers are depicted in order of increasing peak-power profiles. The bar series denoted as average power consumption is the aggregated equivalent of the bars seen previously, in Figure 4.4.

It is interesting to see that ciphers scoring high in the previous test, such as IDEA and LOKI91, display a large difference of roughly 35 *mW* between average and peak power, which can potentially throw the implant designer off track. This difference has to be taken seriously into account if such ciphers are to be employed in an implantable device. That said, IDEA, MISTY1, LOKI91 and RIJNDAEL still occupy the first positions. However, TWOFISH in now inside the top-scoring ciphers and, what is more, it displays the most consistent profile between average and peak power. In terms of our chosen plaintext sizes, and similarly to average power, peak-power profiles present no differences.

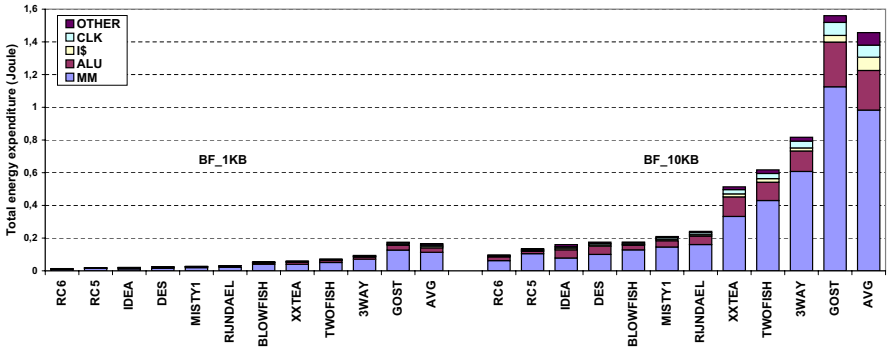


Figure 4.6: *Per-component and total encryption energy costs (in Joules).*

4.4.3.2 Energy expenditure

Apart from the rate at which a cipher consumes energy, i.e. its power consumption, it is important to also investigate the total energy costs incurred for executing the whole cipher. This metric is the total energy expenditure of a cipher and our findings are summarized in Figure 4.6, for both plaintext sizes. SKIPJACK and LOKI91 have been omitted from the plots since they display excessively large energy needs (an order of magnitude larger for LOKI91 than the rest of the algorithms). However, average values include these two algorithms in their calculation to give a complete view.

Knowing the overall energy budget needed for completing a single encryption task is especially important for implantable systems. It directly tells us how much stored energy the given task needs in order to execute and, in effect, what energy amount will be deduced from the battery. It also tells us if e.g. a scheduled encryption and transmission of physiological readouts can take place or not. Given the mission-critical tasks implants perform, it might be preferable at some point to not engage in transmission of (encrypted) data. For instance, it is more important for a pacemaker running low on battery to keep working for an extra couple of days (to allow time for recharging or servicing) than to transmit ECG readouts to its inquiring host once and then power down.

In terms of energy distribution in the various processor components, we can again see that the MM, ALU, CLK and I$ are the most demanding ones. However, Figure 4.6 tells a completely different story for the energy sparingness of the profiled ciphers. RC6 and RC5 have climbed in the first positions of the ranking, becoming the most energy-efficient ciphers. IDEA and MISTY1 follow with RIJNDAEL and BLOWFISH contesting for the fifth position across

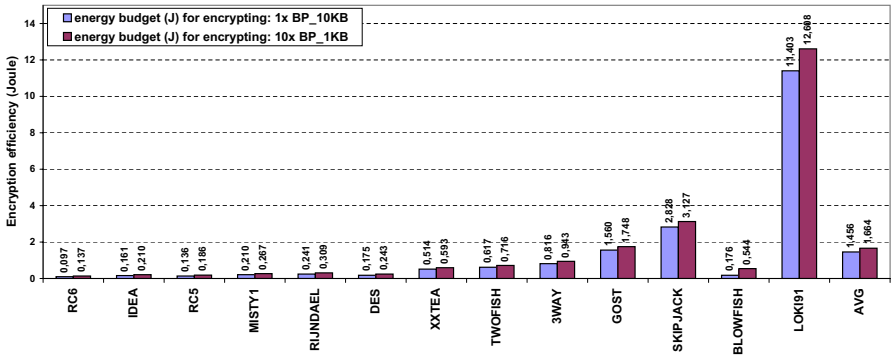


Figure 4.7: *Computation overhead of ciphers manifested as energy penalty (in Joules) when encrypting one 10-KB and ten 1-KB plaintexts.*

the two different plaintexts. Clearly, MISTY1 and RIJNDAEL perform better when smaller plaintext sizes are considered. Conversely, BLOWFISH favors larger sizes. Further, it is surprising that XXTEA is not among the best-scoring ciphers since it is considered a relatively light-weight algorithm.

A last observation in this subsection is that energy budget *does not scale linearly* with plaintext size for most of the ciphers. The cost of encrypting a 10-KB workload as opposed to that of successively encrypting 10 1-KB workloads is 14% smaller, in an overall. The reason for that difference again is the overhead penalty paid during initialization of the encryption algorithms (e.g. key setup). As our simulations have revealed, other factors also contributing to this penalty are the increased fetch- and data-stalls that are reduced over the execution time of a cipher as cache entries get filled, etc.. However, this penalty is not similar across the various ciphers. In Figure 4.7, the energy budgets for encrypting one 10-KB workload and 10 consecutive 1-KB workloads are plotted. The ciphers are ranked in order of increasing difference between the two budgets, i.e. in order of increasing penalty. RC6, IDEA, RC5, MISTY1 and RIJNDAEL are still in the first positions, incurring small penalties but TWOFISH has fallen near the bottom of the ranking, due to introducing a significant energy penalty. This secondary metric of energy is interesting because it indirectly gives a measure of computational efficiency of the various ciphers.

4.4.3.3 Encryption rate

Another metric we use in our profiling study of block ciphers is their encryption rate. In Figure 4.8 encryption rates in KB/sec are reported for 1-KB and

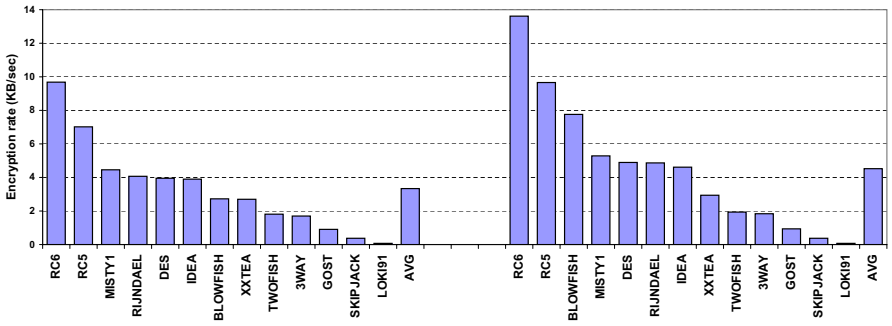


Figure 4.8: *Encryption rate (in KB/sec).*

10-KB plaintexts. RC6, RC5, MISTY1, RIJNDAEL and BLOWFISH score the highest on this metric, with RC6 and RC5 being by far the fastest ciphers. In fact, and contrary to the rest of the ciphers, RC6 and RC5 achieve impressive encryption-rate improvements with increasing plaintext size. The rate of BLOWFISH appears also to benefit largely from a larger plaintext size. In an overall, all ciphers seem to benefit from larger plaintexts: from $3.34\ KB/sec$ for the 1-KB data, the average rate boosts to $4.52\ KB/sec$. The reasons for this are the same as the ones previously mentioned concerning the energy penalty. They are related to the cipher-key initialization phase as well as the cold start of the processor itself.

For the targeted implant applications, our primary concern is to preserve power consumption at low levels. This means that we are not seeking the fastest performing cipher but, rather, one which is fast enough to cover our needs. As can be seen in Table 4.2, the biological data we used as plaintext features a relatively high (in this context) sampling rate of $4.86\ KB/sec$ for the 1-KB and $5.22\ KB/sec$ for the 10-KB workload. In our 2-MHz simulated processor, only ciphers RC6 and RC5 manage to sustain the required sampling rate. The cost paid is that both ciphers display a relatively high power profile ($93\ mW$ to $100\ mW$) as seen in Section 4.4.3.1.

4.4.3.4 Executable-binary size

In order to give a measure of proportion to our profiling study, it is useful to also report on the size of the encryption-algorithm executables, as a measure of program-memory needs. Since XScale supports the ARM ISA and, accordingly, XTREM is based on a modified version of SimpleScalar/ARM, executables have been built with a custom-modified version the GNU ARM-

encryption algorithm	size (KB)
XXTEA	11.2
3WAY	11.2
LOKI91	11.3
RC6	11.4
RC5	11.4
GOST	12.2
SKIPJACK	12.2
IDEA	13.4
DES	14.6
BLOWFISH	15.3
MISTY1	18.8
TWOFISH	22.2
RIJNDAEL	37.0

Table 4.6: *Program sizes (in KB) of the encryption algorithms.*

GCC v2.95.2 cross-compiler. Due to these custom modifications, we discovered that the binaries generated by this ARM-GCC all have the same size. [3]. Furthermore, executables have been statically linked (this is an ARM requirement) and, therefore, are expected to be somewhat larger in size than their e.g. 8086-architecture counterparts. Optimization level 2 (-O2 flag) has been used instead of level 3 (-O3 flag). It could possibly make faster code but the applications that benefit from it are very few, usually image and video decoders. However it has a side effect: it always generates a larger binary sizes. Since video/audio applications are not included in our workloads and we try to avoid large binaries as much as possible, O2 was selected as a proper optimization level. In Table 4.6, the code complexities of the selected encryption algorithms are shown in ascending order.

Obviously, results shown in the table are implementation-dependent and should be considered with caution. However, as we mentioned also in Section 4.4.2.3, many different algorithms have been based on the same software architecture, built by the same author(s). Therefore, the difference in sizes (not the actual sizes themselves), can give an indication of the difference in program-memory needs, regardless of the underlying implementations. Best scoring algorithms in this case are XXTEA, 3WAY, LOKI91, RC6 and RC5.

[3] To cope with this problem, a standard version (namely, ARM-GCC v4.1.2) has also been invoked here to calculate the actual binary sizes of the various algorithms.

encryption algorithm	key size (bits)	Security margin
GOST	256	2243
BLOWFISH	128	2076
IDEA	128	2076
RC5	128	2076
XXTEA	128	2076
MISTY1	128	2076
RC6	128	2076
TWOFISH	128	2076
RIJNDAEL	128	2076
3WAY	96	2034
SKIPJACK	80	2013
LOKI91	64	1992
DES	56	1982

Table 4.7: *Security margins of the encryption algorithms.*

4.4.3.5 Security margin

Since we are evaluating encryption ciphers, a last, suitable metric of our comparative study is the security level provided by each cipher. According to Lenstra and Verheul [82], a cryptosystem can be assumed to be secure only if it is considered to be sufficiently infeasible to mount a successful attack. Unfortunately, it is hard to quantify what precisely is meant by "sufficiently infeasible". To cope with this known issue, we adopt the widely used *security margin* metric, proposed also by Lenstra and Verheul, which is defined as the year until which a user was willing to trust the DES cipher.

According to this definition, if an attacker could afford C_{DES} computations in 1982, sufficient to break DES, and can afford C_X computations in year y ($y > 1982$), sufficient to break cipher X, then the security of cipher X in year y is computationally equivalent to the security of DES in 1982, or in other words, the security margin of cipher X is y. Since DES was standardized in 1977 and set for review in 1982, the year 1982 is used as the baseline. If the best known attack against a cipher with key length k is exhaustive key search, y can be calculated according to:

$$y = 1982 + \frac{30}{23} * (k - 56).$$

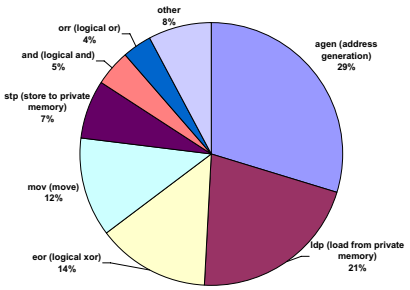
Security margins for our studied ciphers are shown in Table 4.7 in descending order. Based on the previous discussion, all algorithms except for LOKI91 and (of course) DES are secure. Also, SKIPJACK, although secure, displays the

average power consumption	peak power consumption	total energy cost	encryption efficiency	encryption rate	program-code size
IDEA	IDEA	RC6	RC6	RC6	XXTEA
LOKI91	MISTY1	RC5	IDEA	RC5	3WAY
SKIPJACK	LOKI91	IDEA	RC5	MISTY1	LOKI91
MISTY1	TWOFISH	MISTY1	MISTY1	RIJNDAEL	RC6
RIJNDAEL	RIJNDAEL	BLOWFISH	RIJNDAEL	BLOWFISH	RC5

Table 4.8: *Five best-performing encryption algorithms (in descending order of performance).*

ARM instruction	Equiv. ARM microcode
stmdb r13!,r4-r8,r10-r15	agen tmp1,r13,0 agen tmp0,tmp1,-16 stp r11,[tmp0] agen r13,r13,-16 agen tmp0,tmp1,-12 stp r12,[tmp0] agen tmp0,tmp1,-8 stp r14,[tmp0] agen tmp0,tmp1,-4 stp r15,[tmp0]

(a) *Sample ARM instruction which stores registers in the stack pointed to by R13 and equivalent ARM µop sequence.*



(b) *µop mix and frequencies for MISTY1 operating on the 1-KB BP plaintext.*

Figure 4.9: *ARM microcode representation and MISTY1 µop frequencies*

next shortest security margin. Conclusively, LOKI91 and SKIPJACK - if it does not appear to rank high in the rest of the metrics - will be left out from our final selection process.

4.4.4 Results & discussion

To summarize our analysis results, we present in Table 4.8 the 5 best-performing algorithms per profiled metric (except for the security-margin metric). MISTY1 appears in 5 out of 6 metrics in the above table. IDEA, RIJNDAEL, RC6 and RC5 follow, each with 4 occurrences in the table. However, IDEA performs consistently better than RIJNDAEL with the exception of encryption rate. Besides, RIJNDAEL scores almost always last in the ranking among ciphers with 4 occurrences. Last, RC6 scores always better than RC5. LOKI91 has 3 occurrences in the table but is, in any case, dismissed due to its – now – insecure nature.

IDEA μ**op**	**percentage**	**RC6** μ**op**	**percentage**
mov (move)	29%	**agen (address generation)**	25%
agen (address generation)	18%	**mov (move)**	15%
ldp (load from private memory)	12%	**ldp (load from private memory)**	15%
b (unconditional branch)	9%	**stp (store to private memory)**	8%
add (add)	7%	**add (add)**	8%
cmp (compare)	6%	**b (unconditional branch)**	5%
stp (store to private memory)	5%	**eor (logical xor)**	4%
orr (logical or)	4%	**sub (subtract)**	4%
other	10%	**rsb (reverse subtract)**	4%
		other	11%

Table 4.9: *μop mix and frequencies for IDEA and RC6 operating on the 1-KB BP plaintext.*

Conclusively, from the above findings, MISTY1 is the most promising cipher according to our imposed metrics; thus, we take a closer look at its underlying instruction mix. Figure 4.9b illustrates the type and frequency of instructions executed for encrypting the 1-KB BP plaintext with the MISTY1 cipher. XTREM, which is based on SimpleScalar, implements ARM instructions through an internal microcode representation (simply referred to as μops hereon). We included μop statistics rather than the actual ARM instructions because the μops can better capture the workings of the underlying architecture. For instance, a single ARM command to store multiple registers to the stack pointed to by R13, breaks down to a number of more elementary μops (see Figure 4.9a).

Going back to Figure 4.9b, we readily observe that the *address-generation* operation (agen) – which, in essence, is a standard arithmetic operation – is by far the most common and, although it is specific to ARM-based microarchitectures, it reveals the importance of implementing an efficient address-generation mechanism in the envisioned processor. Execution is also heavily dominated by load/store (stp, ldp) operations, logic operations (eor, and, orr) and register-to-register copy operations (mov). This mix motivates us towards efficient implementation of loads/stores, moves and logic operations in terms of power consumption and execution speed.

By investigating also the second and the third best ciphers, i.e. IDEA and RC6, we accumulate the statistics, seen in Table 4.9. The mixes in this case favor, too, address-generation, load/store and move operations but logic operations to a smaller extent, compared to the MISTY1 case. However, they both display high percentages of arithmetic (add, sub, rsb, cmp) and branch

(b) operations, contrary to MISTY1. Given that MISTY1 scores high in most metrics of our profiling study, optimizing our architecture for the more focused MISTY1 μop mix is considered the best option. Also, since address generation is a particularly frequent μop, mechanisms for speeding up this specific arithmetic operation might probably prove highly beneficial in the SiMS processor. However, one should keep in mind that "agen" is a μop specific to the ARM microarchitecture and might, therefore, bring diminished benefits in a microarchitecture of different approach.

4.5 Profiling of compression algorithms

Similarly to the study previously performed on encryption algorithms, in the following sections, we profile various popular lossless-compression algorithms against suitable metrics. Then, we select the ones with the best characteristics for the targeted application domain and again investigate their respective instruction frequencies and mixes, useful for offering insights on the design and implementation of the targeted processor.

4.5.1 Selection criteria of compression algorithms

As previously discussed, implants periodically record or generate signals which they store in an on-board memory and selectively transmit wirelessly to an external monitoring station. This pattern of behavior indicates that outbound biological-data traffic almost always dominates inbound traffic.

It has been measured [11] that putting a single bit on the air for transmission consumes more energy than a 1,000 32-bit computation (i.e. "add") operations[4]. It, then, becomes apparent that performing a 1,000 extra operations for compressing data by even 1 bit saves overall energy expenditure.

Since the transmissions from the implant to the outside world are expected to be more than the receptions, it becomes apparent that – in order to save over implant energy – information needs to be compressed before transmission. Therefore, in the context of implants, the most important aspect of the pair data compression-expansion is the former, thus this work deals only with the *compression aspect* of the studied compression algorithms. This choice affects the findings of the study since many compression schemes display un-

[4]In [11], it has been roughly measured that transmitting 1 bit of information is approx. equivalent to 485–1267 add instructions, in terms of energy.

feature	value
ISA	32-bit ARMv5TE-compatible
Pipeline depth	7/8-stage, super-pipelined
Datapath width	32-bit
RF size	16 registers
Issue policy	in-order
Instr.window	single-instruction
I-Cache, L1	32B, 1-entry, 1-cc hit/170-cc miss lat.
D-Cache, L1	32B, 1-entry, 1-cc hit/170-cc miss lat.
BTB	2-entry direct-mapped
TLB	1-entry
Branch Predictor	2-bit Bimodal
Write Buffer	2-entry
Fill Buffer	2-entry
Mem. bus width	1 Byte
INT/FP ALUs	1/1
Clock freq.	2 MHz
Implem. tech.	0.18 μm @ 1.5 Volt

Table 4.10: *XTREM configuration for compression profiling study.*

balanced compression and expansion complexities; e.g. the compression effort typically is much higher than the expansion effort.

Furthermore, the sensitive nature of biomedical signals dictates that, in the general case, no information can be afforded to be lost or altered during data acquisition, compression and transmission. We are, therefore, inclined to consider solely *lossless compression* to ensure complete information recovery at the receiving end.

4.5.2 Experimental setup

4.5.2.1 Simulator configuration

As discovered in the preceding study on encryption algorithms, XTREM components (with the exception of the MM) scale properly performance- and power-wise. To better match our application field and in lack of better knowledge in the field, many of XTREM's architectural parameters have been cut down or disabled for the following study, in order to better reflect the highly constrained implantable processors. The modified XTREM characteristics are summarized in Table 4.10.

compression algorithm	shorthand name	details
Static-Huffman Coding [101]	huff	Huffman coding with static symbol table
Adaptive-Huffman Coding [101]	ahuff	Huffman coding with adaptive symbol table
Arithmetic Coding, Order-0 [101]	arith	Simple arithmetic coding
Arithmetic Coding, Order-1 [101]	arith1	Order-1 arithmetic coding
Arithmetic Coding, Order-1e [101]	arith1e	Order-1 arithmetic coding with escape characters
LZSS (12-bit sliding window) [101]	lzss	Storer & Szymanski's slightly modified LZ77 version
LZW (fixed 12-bit) [101]	lzw12	LZW with fixed 12-bit symbols
LZW (variable up to 15-bit) [101]	lzw15v	LZW with variable-size symbols, up to 15 bits
Run-Length Encoding [48]	bclrle	Simple run-length encoding
Shannon-Fano [48]	bclsf	–
Finnish [29, 100]	fin	LZ77-variant with 2-character memory window
Splay-Tree Compression [29, 70]	splay	Similar to Huffman encoding, locally adaptive
LZSS w/ Adaptive-Huff. Coding [29]	lzhuf_oku	LZSS with binary-tree symbol table
LZSS w/ Adaptive-Arith. Coding [29]	lzari_oku	–
Urban [100]	urban	High-order arithmetic coder working at the bit level
MiniLZO [103]	mlzo	Light-weight subset of the LZO library (LZ77-variant)
S-LZW [116]	slzw	Memory-constrained modification of LZW for Sensor-nodes

Table 4.11: *Collection of profiled lossless-compression algorithms.*

4.5.2.2 Compression datasets

An overview of selected implant datasets has been provided in Table 4.2. For this study, all input datasets (of both 1-KB and 10-KB sizes) have been used.

Due to the huge amount of data generated during the profiling phase, in the following analysis we only report cumulative figures based on the averaged results across all profiled datasets. That is to say, we do not favor any of the datasets presented in Table 4.2. Further, all reported average values in fact are *median* values unless stated otherwise, since we cannot guarantee normal data distribution in the general case. Last, results have been grouped in two main categories of 1-KB and 10-KB data so as to capture also the variation in behavior when increasing the input size.

4.5.2.3 Compression algorithms

When putting together our collection of compression algorithms, we have made an effort to include sources adhering to similar principles as those fol-

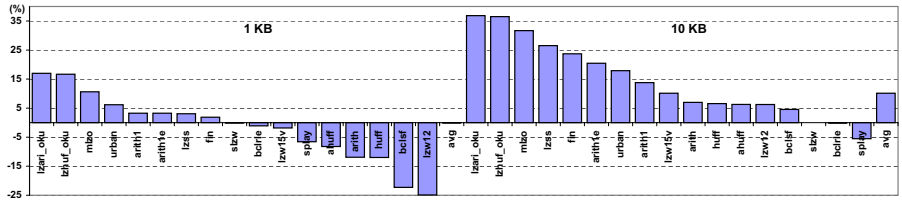


Figure 4.10: *Averaged compression ratios for 1-KB and 10-KB datasets.*

lowed when collecting encryption algorithms; namely: i) large range of lossless data compression techniques and styles, from high-performing to compact flavors; ii) mature implementation code base; iii) various algorithmic complexities; iv) suitability: the XTREM simulator can only handle C and Java sources. Furthermore, in its current version it does not support an OS on top of the simulated hardware, thus prohibiting the use of compression sources - such as the excellent bzip2 algorithm - that require high-level, OS features; and v) availability: all collected algorithms comprise utterly free, published or free under the GNU General Public License sources, readily available to the research community.

The implementation of a given compression algorithm plays as crucial a role for the performance and behavior of the algorithm as its underlying structure. While adhering to the above principles, in order to offer the best possible fairness in our selection process, we have attempted to include algorithms built with the same implementation philosophy (e.g. algorithm suite implemented by the same author(s)) and/or algorithms being top representatives in their category. Table 4.11 summarizes the selected algorithms[5].

4.5.3 Profiling analysis

4.5.3.1 Compression ratio

The first profiled metric to discuss is compression ratio and findings are illustrated in Figure 4.10. For the case of 1-KB data, our compression algorithms perform worse (-0.08% on average) that for the 10-KB data (10.14% on average). For the 1-KB case we actually see an expansion of data, on average. Given that workloads in this case are 10 times smaller, an overall approx. 100% poorer compression is performed. To put it simply, attempting to compress 10 consequent 1-KB readouts results in a compressed output double the size of a

[5] Available online, on the SiMS website: `http://sims.et.tudelft.nl`

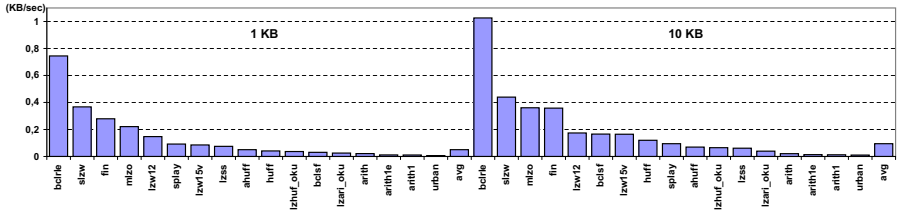


Figure 4.11: *Averaged, average compression rates for 1-KB and 10-KB datasets.*

compressed, single, contiguous 10-KB readout. Clearly, compression of larger files is favored. This claim has to be backed also with energy-expenditure results in order to make it attractive for ULP systems such as implants are. We will address this topic later.

The difference in compression ratios for different workload sizes is justified by the fact that for small inputs, many compression algorithms do not simply have sufficient context to become efficient; symbol tables may not have the time to be filled thus impacting compression efficiency. In short, it is a "cold start" problem. Overall, the most compression-efficient algorithms, as the figures indicate, are *lzari_oku*, *lzhuf_oku* and *mlzo*. *urban* and *arith1* are contesting with *lzss* and *fin* for the 4th and 5th positions, respectively.

4.5.3.2 Compression rate

Another interesting attribute of the compression algorithms is how fast they are able to pack data, i.e. their compression rate. In Figure 4.11 average compression rates in KB/sec are reported. Overall, the average compression rate for 1-KB data is $0.051\ KB/sec$ while for 10-KB data it is $0.095\ KB/sec$, or about double the speed. The reason for this difference is anticipated to be the fact that with 1-KB data, compression algorithms do not have the time to create and traverse excessively large data structures such as the symbol table. For instance, with a typical size of $256\ Bytes$ which is comparable to the input data size of $1\ KB$, the symbol table does not have the time to fill and become efficient. Of course, this has adverse effects on compression. Best-scoring algorithms for this metric are *bclrle*, *slzw*, *fin*, *mlzo* and *lzw12*. *bclrle* achieves by far the most impressive results due its simplistic design but does so at the cost of poor or no compression.

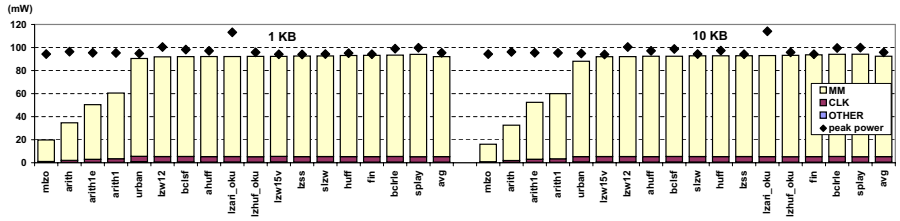


Figure 4.12: *Averaged, average and peak power consumption for 1-KB and 10-KB datasets.*

4.5.3.3 Average & peak power consumption

Average power consumption is an important metric of an algorithm's performance. It reveals the average rate at which the executed algorithm draws energy from the system. An implant battery may have sufficient charge to support a whole compression operation, however, it might not be able to sustain the energy rate needed by the compression algorithm. Another interesting metric in this context is peak power consumption. A battery able to support a compression algorithm with a given average power consumption may be unable to deliver the required output at a given point in time if the algorithm sporadically presents peak power values which are largely deviating from its average power needs. To address both aspects of the profiled algorithms, we have plotted Figure 4.12. The algorithms are ordered in order of increasing average-power profiles. Bars indicate average (overall and per-processor-component) power while black dots indicate peak power.

We can readily see that the memory-manager unit (MM) is the power-hungriest component with a rough 94% fraction of overall power consumed throughout both workload groups. The MM unit is activated each time the core is stalled because of a main-memory instruction or data fetch. Regardless of the aforementioned issue with the non-scaling power model of the MM, a high power consumption in this unit is generally expected for resource-constrained devices with small or totally absent I/D-caches as the ones we consider here. Next follows the CLK consuming about 5% of the overall power.

From the figure we can further discern that average power consumption increases marginally with workload size. In effect, the algorithms' power needs are unaffected by the workload size they operate on. We can also see that most algorithms converge to a consumption threshold of roughly 95 *mW*. We have performed some further tests whereby some of the processor's characteristics have been enhanced, e.g. cache sizes have been increased. In that

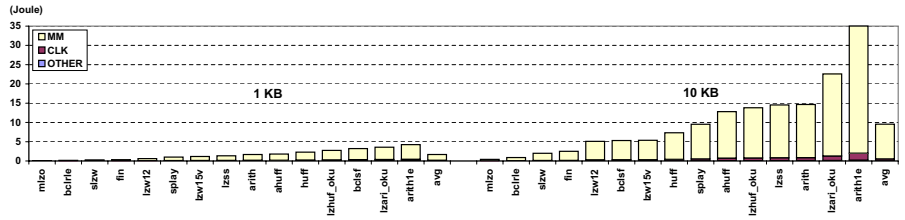


Figure 4.13: *Averaged, total energy expenditure for 1-KB and 10-KB datasets.*

case, a large variation among the power profiles of the various algorithms has resulted. This indicates that the constrained version of the processor we currently use essentially "chokes" the performance of many algorithms forcing them to slow their execution down and, thus, demand less power from the underlying machine. This is a crucial observation since it excludes from selection those algorithms whose performance enhancements will not bring any benefit to a highly resource-constrained, implant processor. Outright best performing algorithms in terms of average power consumption are *mlzo*, *arith*, *arith1e*, *arith1* and *urban*. When peak power consumption is considered, the ranking changes with *lzw15v*, *lzss*, *fin*, *mlzo* and *slzw* scoring best, indicating large deviations between average and peak power figures.

An interesting point to make here is that implantable systems would greatly benefit from power-aware compression techniques. In effect, compression algorithms that dynamically adapt their actual compression speed and/or ratio depending on the amount of energy they spend in a given time interval. When this amount surpasses a preset (or dynamically set) threshold value, they lower their performance to make it back to the threshold. Of course, this presumes a way for the algorithm (thus, software) of tapping into processor (thus, hardware) power figures at run-time. None of the profiled algorithms has such capabilities, yet it would be a crucial adaptation for future ULP systems.

4.5.3.4 Overall energy budget

Knowing the overall energy budget needed for completing a single compression task is important for battery-operated implants. It directly tells us how much stored energy the given task needs in order to execute and, in effect, what stored-energy amount will be deduced from the battery. It also tells us if the compression computation is worth the effort compared to simply transmitting the data uncompressed over the air. Accordingly, in Figure 4.13 averaged, overall energy expenditures for both workload sizes have been plotted. *urban*

bench.	size (KB)	bench.	size (KB)	bench.	size (KB)
fin	10.4	bclrle	15.7	arith1	17.1
splay	12.5	bclsf	15.7	arith1e	17.1
urban	13.5	huff	16.2	lzhuf_oku	17.4
lzw12	13.8	mlzo	16.3	arith	17.4
slzw	14.0	lzw15v	16.7	ahuff	21.5
lzss	14.6	lzari_oku	17.0		

Table 4.12: *Compression algorithms' program sizes.*

and *arith1* display very large energy costs and have, thus, been omitted from the plots to give better resolution for the rest of the algorithms.

From Figure 4.13, we can readily observe that the energy budget *does not scale linearly* with workload size. The cost of compressing one 10-KB workload (`9.541` *J*) as opposed to that of successively compressing 10 1-KB workloads (`1.684` *J* for one) is about 55% smaller. This agrees also with our compression-ratio results; that is, rarer compression of larger input data is energy- and compression-wise preferable to frequent compression of smaller input data. Agreeing with the previous discussion on power, we can further see that the MM and CLK components indeed are the overall most energy-consuming parts of the processor. The best performing algorithms in this case are *mlzo*, *bclrle*, *slzw*, *fin* and *lzw12* and they preserve their ranking for both workload sizes. Interestingly, with the exception of *mlzo*, these are not the same algorithms as the top-ranking ones in terms of average power consumption, as one might expect. The reason for this difference lies in the actual algorithm execution times. An algorithm might consume little power on average but might do so for a disproportionately large amount of time, thus canceling all benefits of its low-power nature. For instance, *arith* consumes only `32.58` *mW* on average while compressing a 10-KB workload but it completes its task in `456.63` *sec* on average while the overall average compression time for 10-KB workloads is only `86.28` *sec*. Hence, its excessive energy budget and resulting poor ranking.

4.5.3.5 Executable-binary size

A last metric we evaluate is the binary size of the algorithms' executables, as a measure of program-memory needs. Executables have been built with the GNU ARM-GCC v4.1.2 cross-compiler and optimization level O2. Furthermore, executables have been statically linked (this is an ARM require-

ratio	avg. rate	avg. power	peak power	total energy	code size
lzari_oku	bclrle	mlzo	lzss	mlzo	fin
lzhuf_oku	slzw	arith	lzw15v	bclrle	splay
mlzo	fin	arith1e	fin	slzw	urban
urban	mlzo	arith1	mlzo	fin	lzw12
arith1	lzw12	urban	slzw	lzw12	slzw
lzari_oku	bclrle	mlzo	lzw15v	mlzo	fin
lzhuf_oku	slzw	arith	lzss	bclrle	splay
mlzo	mlzo	arith1e	fin	slzw	urban
lzss	fin	arith1	mlzo	fin	lzw12
fin	lzw12	urban	slzw	lzw12	slzw

Table 4.13: *Five best-performing compression algorithms in descending order (top: 1-KB, bottom: 10-KB).*

ment) and, therefore, are expected to be somewhat larger in size than their dynamically linked counterparts. In Table 4.12, the code complexities of the selected compression algorithms are shown in ascending order. Obviously, results shown in the table are heavily implementation-dependent and should be considered with caution. However, as mentioned in Section 4.5.2, many different algorithms have been based on the same software architecture, built by the same author(s). Therefore, the difference in sizes can give an indication of the program-memory needs, regardless of the underlying implementations. Best scoring algorithms in this case are *fin*, *splay*, *urban*, *lzw12* and *slzw*.

4.5.4 Results & discussion

To summarize our analysis results, we present in Table 4.13 the 5 best-performing algorithms on each one of our profiled metrics, for both workload sizes. The undisputed winner is *mlzo*, followed by *fin* and *slzw*. Accordingly, we take a closer look at the underlying instruction mix of *mlzo*.

As discussed in the analysis of encryption algorithms, the XTREM simulator internally breaks up executed ARM instructions to "μops", by design. This quirk in fact is useful to us since it allows us to capture microarchitectural details at the smallest granularity possible. We modified the simulator to be able to capture these "μop" dynamic traces. In Table 4.14, the on-average most frequent ($> 5\%$) μops for both workload sizes are listed. The *address-generation* ("agen") μop is by far the most common and, although it is specific to ARM-based microarchitectures, implementing an efficient address-generation mechanism in the envisioned processor might benefit performance and power con-

μop	avg(1KB)	uop	avg(10KB)
agen	30.00%	agen	26.88%
ldp	19.89%	ldp	20.57%
b	9.94%	b	12.18%
cmp	8.53%	cmp	9.65%
stp	8.29%	add	7.39%
mov	5.90%	stp	5.91%
add	5.66%	eor	5.77%

Table 4.14: *Popular* mlzo *μop frequencies.*

```
repeat for all consecutive instruction triplets of the program {

  let instr1, instr2, instr3 be 3 new consecutive instructions.

  if (instr2.src_reg1 == instr1.dest_reg) or
     (instr2.src_reg2 == instr1.dest_reg)
    then instr2 is dependent on instr1 (pair).

  if (instr3.src_reg1 == instr1 dest_reg) or
     (instr3.src_reg2 == instr1.dest_reg)
    then also instr3 is dependent on instr1 (triplet).
} end
```

Table 4.15: *Instruction-dependency algorithm.*

μop	pairs	/ triplets	avg(1KB)	avg(10KB)
and	eor	-	14%	17%
eor	eor	-	8%	5%
and	eor	and	-	7%
eor	cmp	-	-	6%
beq	add	add	-	6%

Table 4.16: *Popular* mlzo *μop pairs and triplets.*

sumption significantly. *Loads* ("ldp") follow in frequency, justifying the previously observed large power component of the MM unit and hinting towards a power-efficient MM design, if at all present. *Branch/jump* ("b") and *compare* ("cmp") instructions follow and expectedly have similar occurrence frequencies. They indicate that even small optimizations in the compare-and-branch mechanism will improve power and performance significantly.

Lastly, the XTREM simulator has been further modified to also collect pairs

and triplets of data-dependent instructions (or "μops"[6]) during execution time. Data-dependent instructions have been defined according to the simple algorithm shown in Table 4.15. In effect, we are scanning for all data-dependent dynamic instructions of the program and are interested in the exact nature of those pairs or triplets of dependent instructions present.

[6]The terms "instruction" and "μop" are used interchangeably in this text to signify the same concept.

We report Table 4.16 listing popular dynamic instruction pairs/triplets during *mlzo* execution for both workload sizes. Instruction pairs or triplets are consecutive μops whereby data generated by the first μop is consumed by the second and/or third μop; i.e. whereby data dependencies occur. The table reveals that by far the most popular pair is "and-eor" (eor: exclusive or) followed by "eor-eor". We, thus, get a clear indication that data-forwarding in the logical-operation part of the ALU, interlock-collapsing-ALU techniques [141] or other (micro)architectural optimizations will significantly benefit the implant processor. Further, the "and-eor-and" triplet falls in the above category of optimizations. However, the "eor-cmp" and "beq-add-add" combinations relate also to the previous discussion on optimizing the compare-and-branch subsystem of the processor. Last but not least, all above observations on μop frequencies can give clear directions as to which instructions should be explicitly implemented in hardware and which ones can be afforded to be implemented in software (compiler-side conversion).

4.6 ImpBench: A novel benchmark suite for implants

In Section 4.2 we have detailed some of what we consider the most popular workloads for implant applications in the years to come. Based on these elaborations, in Sections 4.4 and 4.5, we have performed profiling studies on the most suitable encryption (*MISTY1* and *RC6*) and compression (*MiniLZO* and *Finnish*) algorithms, respectively, for running on implantable devices. For the two remaining workload categories, i.e. data-integrity algorithms and real implant applications, we have taken a different approach.

For the data-integrity category – effectively belonging to the vast and exhaustively studied field of *error-detecting and -correcting codes* – we have dimmed it more realistic in terms of time to adopt the few but available algorithms already employed in biomedical implants before: Implant designs by Wang L. et al. [145] and Eggers et al. [37] indicate the use of simple checksum and CRC-8/CRC-16/CRC-32 codes for protecting information in the implant.

Real implant applications, on the contrary, are largely unavailable in related literature. We have set out communicating with all authors of the surveyed papers reported in Chapter 2. We got few (positive) responses out of which even fewer ones presented usable code. In the end, we have come up with two application source codes (in C) based on the works of Wouters et al. [152] and of Cross et al. [25]. The former code is our own software version of a motion-detection algorithm implemented in the implant by Wouters et al.. The latter

code is our modified version (for proper execution in XTREM) of the exact embedded-C application code, released directly by the authors Cross et al. As will be discussed in the following sections, these application codes eventually constitute two synthetic and very diverse applications, as was our original goal.

4.6.1 The need for a new benchmark suite

We have already shown that the list of potential implant applications is constantly expanding and the number of software-based implant solutions is increasing. The need for a formal, standardized way of designing and evaluating future implant architectures becomes apparent and, to this end, a collection of carefully selected benchmark programs is needed.

While in the areas of general-purpose computing, multimedia and networking, to name a few, research has relied on well-established workload-characterization suites such as the SPEC benchmark suite [128] for optimizing the underlying hardware, this has not been the case in the area of implant-processor design. To address this need, we have developed the ImpBench (*Imp*lant-*Bench*mark) suite. Through ImpBench we set the following goals:

- Identify a common subset of programs representative of the workloads of existing and emerging implantable systems;
- Propose self-contained programs written in a popular, HLL so as to allow for easy porting to new implant cores under evaluation;
- Propose a free benchmark suite to the research community;
- Verify the uniqueness and, thus, usefulness of ImpBench as compared to other existing benchmark suites.

4.6.2 The ImpBench components

Even though in the previous sections we have detailed the most prominent implant characteristics, such devices have always been and, by nature, will be serving a wide variety of applications. This makes the task of identifying a representative workload set a tough one. ImpBench is expected to be a continuously evolving and updated tool; still, we are confident that we have correctly identified a common subset of programs essential for all current and future implantable systems.

Compression	Encryption	Data integrity	Real applications
miniLZO [103]	MISTY1 [79]	checksum [16]	motion [152]
Finnish [29]	RC6 [79]	CRC32 [19]	DMU [25]

Table 4.17: *ImpBench components.*

To draw a clear structure of our proposed benchmark programs, we have grouped them in four distinct categories of two programs each: *lossless data compression*, *symmetric-key encryption*, *data-integrity* and synthetic programs (what we call henceforth *real applications*). The benchmarks as summarized in Table 4.17 and are as follows:

i. **miniLZO**: MiniLZO is a light-weight subset of the LZO library (LZ77-variant). LZO is a data compression library suitable for data de-/compression in real-time, i.e. it favors speed over compression ratio. LZO is written in ANSI C and is designed to be portable across platforms. MiniLZO implements the LZO1X-1 compressor and both the standard and safe LZO1X decompressor.

ii. **Finnish**: This is a C version of the Finnish submission to the Dr. Dobbs compression contest. It is considered to be one of the fastest DOS compressors and is, in fact, a LZ77-variant, its functionality based on a 2-character memory window.

iii. **MISTY1**: MISTY1 is one of the CRYPTREC-recommended 64-bit ciphers and is the predecessor of KASUMI, the 3GPP-endorsed encryption algorithm. MISTY1 is designed for high-speed implementations on hardware as well as software platforms by using only logical operations and table lookups. MISTY1 is a royalty-free open standard documented in RFC2994 [104] and is considered secure with full 8 rounds.

iv. **RC6**: RC6 is a parameterized cipher and has a small code size. RC6 is one of the five finalists that competed in the AES challenge and has reasonable performance. Further, Slijepcevic et al. [126] selected RC6 as the algorithm of choice for WSNs. RC6-32/20/16 with 20 rounds is considered secure.

v. **checksum**: The checksum is an error-detecting code that is mainly used in network protocols (e.g. IP and TCP header checksum). The checksum is calculated by adding the bytes of the data, adding the carry bits to the least

significant bytes and then getting the two's complement of the results. The main advantage of the checksum code is that it can be easily implemented using an adder. The main disadvantage is that it cannot detect some types of errors (e.g. reordering the data bytes). In the proposed benchmark, a 16-bit checksum code has been selected which is the most common type used for telecommunications protocols.

vi. **CRC32**: The Cyclic-Redundancy Check (CRC) is an error-detecting code that is based on polynomial division. The main advantage of the CRC code is its simple implementation in hardware, since the polynomial division can be implemented using a shift register and XOR gates. In the proposed benchmark, the 32-degree polynomial[7] specified in the Ethernet and ATM Adaptation Layer 5 (AAL-5) protocol standards has been selected (same as in NetBench).

vii. **motion**: This is a synthetic benchmark based on the algorithm described in the work of Wouters et al. [152]. It is a motion-detection algorithm for the movement of animals. In this algorithm, the degree of activity is actually monitored rather than the exact value of the amplitude of the activity signal. That is, the percentage of samples above a set threshold value in a given monitoring window. In effect, this motion-detection algorithm is a smart, efficient, data-reduction algorithm.

viii. **DMU**: This is a synthetic benchmark based on the system described in the work of Cross et al. [25]. It simulates a drug-delivery & monitoring unit (DMU). This program does (and can) not simulate all real-time time aspects of the actual (interrupt-driven) system, such as sensor/actuator-specific control, low-level functionality, transceiver operation and so on. Nonetheless, the emphasis here is on the operations performed by the implant core in response to external and internal events (i.e. interrupts). A realistic model has been built imitating the real system very closely.

As explained in the preceding profiling studies, lossless as opposed to lossy compression algorithms have been included since information deterioration is not an option for implant applications. Also, symmetric- as opposed to asymmetric-encryption algorithms have been included since they characterize better the operational profile of implants. The checksum error-detecting code has been selected for its minimal overhead and effectiveness (it has been

[7] CRC32 generator polynomial: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$.

feature	value
ISA	32-bit ARMv5TE-compatible
Pipeline depth	7/8-stage, super-pipelined
Datapath width	32-bit
RF size	16 registers
Issue policy / Instr.window	in-order / single-instruction
I-Cache, L1	32KB, direct-mapped (1cc-hit/170cc-miss lat.)
D-Cache, L1	32KB, direct-mapped (1cc-hit/170cc-miss lat.)
TLB	1-entry fully-associative
BTB	2-entry direct-mapped
Branch Predictor	2-bit Bimodal (32-entry RAS)
Write Buffer / Fill Buffer	2-entry / 2-entry
Mem. port no / bus width	1 port / 1 Byte
INT/FP ALUs	1/1
Clock frequency	2 MHz
Implem. tech.	0.18 μm @ 1.5 Volt

Table 4.18: *XTREM configuration for ImpBench evaluation.*

used in implantable systems time and again) while CRC32 has already been implemented in various light-weight network protocols including the energy-scavenging ZigBee. Lastly, we have implemented both real applications (*motion* and *dmu*) after extensively investigating the diverse field of implant applications and consider them capable of capturing commonly met operations in contemporary and future implants. Suitable datasets representing biological content have been used to feed all benchmarks. Particularly for the *dmu* benchmark, *actual field data* have been used in order to capture the exact behavior of the simulated implantable system.

By including pairs of different algorithms performing similar functionality in ImpBench, we attempt to offer some benchmarking diversity able to capture different aspects of a new system when evaluated against the suite. This diversity will be further illustrated in Section 4.6.4.

4.6.3 Experimental setup

For evaluating the uniqueness and usefulness of ImpBench, we have chosen to compare it against a number of benchmark programs extracted from **MiBench**. MiBench, rather than SPEC, MediaBench or other benchmarks suites (discussed in Section 3.3), appears to be the most closely pertinent – in terms of workloads – to the application field we are targeting. In order to perform fair comparisons between the two suits across various metrics, we had to run both benchmark collections in a suitable profiling platform.

Profiling has been based on XTREM. The modified XTREM characteristics used in this experiment are summarized in Table 4.18. The reasons for the current XTREM configuration will be explained through the course of the following analysis.

4.6.4 Benchmark characterization

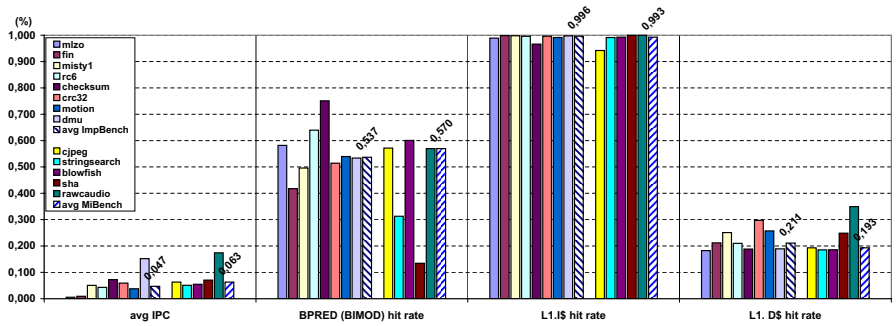
Since the MiBench suite spans a wide range of application fields, we have selected a small but representative subset from the ones most related to our own field. Selection has also been based on porting issues since not all MiBench programs could be successfully compiled on our bare-metal (i.e. no OS-support) simulator. From the "Consumer" category *jpeg* has been been chosen, from the "Office" category *stringsearch*, from the "Network" category *blowfish*, from the "Security" category *SHA* and from the "Telecomm." category *ADPCM enc.*. For all profiled benchmarks, only the compression part of compression algorithms and the encryption part of cryptographic algorithms have been considered since they are the most computationally demanding aspects and/or are the most commonly executed from the point of view of implants.

The goal of this phase is to empirically test whether the ImpBench suite is *quantitatively* different from the MiBench suite, each operating on its own representative datasets (*inter-benchmark* variation). We also wish to point out the variety in behavior offered by the two alternative flavors in each one of the four ImpBench categories (*intra-benchmark* variation). Most results indicate relative values, that is, ratios.

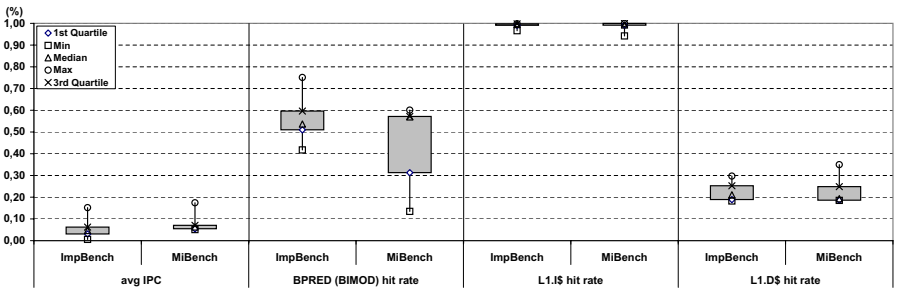
4.6.5 Performance, caches and branch prediction

The first characteristic we explore is benchmark performance measured as the Instructions Per Cycle (IPC) of each benchmark. Since IPC depends also on the cache performance and the efficiency of the branch-prediction unit, we include such results here, as well. Accordingly, in Figure 4.14, overall average IPC, L1 I-cache and D-cache hit rates and branch-prediction rates are depicted.

As can be seen from Figure 4.14a, ImpBench programs achieve on average a lower IPC (`0.047`) than the MiBench ones (`0.063`); yet, both IPCs are expectedly low. To elaborate, in order to closely model real implantable processors, the XTREM simulator has been modified to such a degree that all tasks running on it are effectively "choked" by the limited resources left on it. That is, the intrinsic performance of many tasks is capped by the maximum performance the



(a) *Per-benchmark, average values.*



(b) *Box-and-whiskers plot.*

Figure 4.14: *IPCs, I-/D-cache hit rates and branch-prediction rates*

simulated processor can deliver. This is reflected in the limited IPCs observed for both benchmark suites. However, Figure 4.14b captures a more prominent difference between the two suits. Although both suite distributions are skewed closer to their minimum values, ImpBench programs display a wider dispersion as can be seen by the box sizes formed by the 1st and 3rd quartile (i.e. the middle 50% of the values).

In terms of intra-benchmark variation, MiBench's *rawcaudio* (ADPCM encoding) and *sha* perform apparently better than most of the ImpBench programs while *stringsearch* is scoring the lowest in MiBench and even low for many ImpBench programs. The two compression algorithms of ImpBench, although varied, display by far the poorest performance across all benchmarks, seemingly impacted by the limited D-cache size, as will be discussed later. For the encryption algorithms, *misty1* appears to perform better than *rc6* while, for the data-integrity algorithms, *checksum*'s simpler structure clearly outperforms *crc32*. Last, *motion*, of the real applications, although simpler, performs

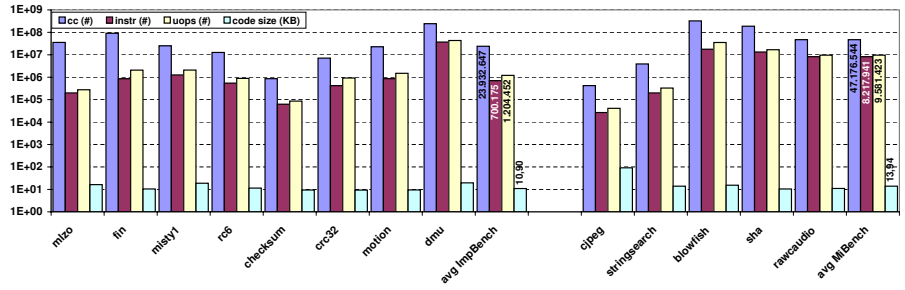
significantly worse than *dmu* contrary to which *motion* displays a higher ratio of I/O- over control- or data-intensive operations. In effect, it reads successive data from a file (representing motion-sensor readouts), compares them against a preset motion threshold value and writes an activity factor to an output file (representing a data-logging memory).

As shown in the section on XTREM configuration, above, a Bimodal branch-prediction scheme has been used with a mere 2-entry table, the reasons being: a) to reflect the constrained nature of an implant processor and, b) to isolate the dynamic behavior of the various benchmarks. Referring back to Figure 4.14, we conclude that overall branch-prediction (BPRED) hit rates are similar for both suites. However, contrary to the observed IPCs, the MiBench programs display a significantly wider dispersion of BPRED values than ImpBench. Further, the MiBench values are strongly skewed towards the maximum value whereas ImpBench values present a distribution closer to the Gaussian. In effect, ImpBench programs present a slightly less predictable but overall more consistent dynamic behavior among them and it is interesting to note that the range (i.e. max-min) of ImpBench BPRED values (`0.333`) is quite smaller than that of MiBench ones (`0.466`).

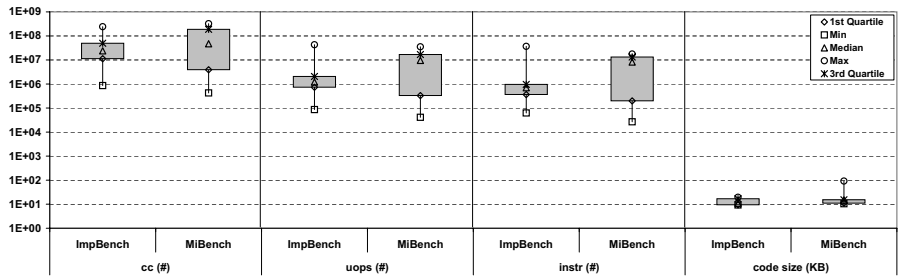
Besides, *fin* achieves a worse BPRED rate than *mlzo* and the worst overall for ImpBench programs but, still,$x3$ better than the worst MiBench program (*sha*). In terms of intra-variation, encryption and data-integrity algorithms also vary largely in behavior while the real applications display similar profiles.

The selected I-cache configuration and the intrinsic behavior of the programs has yielded essentially miss-free cache operation, as can be seen in Figure 4.14a. This behavior is observed in the MiBench programs as well, featuring a marginally smaller I-cache hit rate. Figure 4.14b indicates that, in this case, value dispersion is extremely low with a slight skew towards the maximum value, for both suites. Combined, the two figures tell us that in terms of I-cache behavior, there is no significant difference between the two suites.

The D-cache, on the other hand exhibits a different hit-rate behavior. As seen in Figure 4.14a, ImpBench programs feature an overall `0.211` miss rate, quite higher than its `0.193` MiBench counterpart, but both much lower than the I-cache hit rates witnessed previously. Figure 4.14b further reveals that the dispersion of values is moderate for both suites, with ImpBench being marginally larger. Yet, its distribution is again closer to the Gaussian than that of MiBench whose values are clearly skewed towards the minimum. A last observation to make at this point is that the lower hit rates of the D-cache, as compared to the I-cache, reveal a strong data-intensive nature of the biomedical applications.



(a) *Per-benchmark, average values.*



(b) *Box-and-whiskers plot.*

Figure 4.15: *Static code size (in KB) and dynamic code size (instruction and clock-cycle count).*

4.6.5.1 Dynamic & static benchmark size

We, next, delve into the differences between the two benchmark suites in terms of static and dynamic program size. From Figure 4.15 we can readily observe that, in an overall, ImpBench programs feature a moderately smaller static size, about `10.9` *KB* compared to the `13.94` *KB* of the MiBench programs but their sizes are somewhat more dispersed. *cjpeg* displays the overall largest static size while *crc32* the overall smallest one. In terms of intra-variation, *fin* is smaller than *mlzo*, *rc6* is smaller than *misty1* while *checksum* is slightly larger than *crc32*. *dmu* is much larger than *motion* which is to be expected since the former implements a much larger and more complex application. We can also notice that across the compression and encryption categories, the algorithms which perform better, do so at an increased code size.

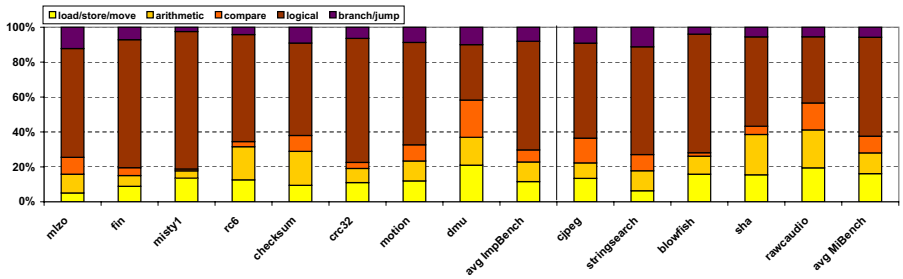
In terms of dynamic behavior, which depends on the input datasets used as well as on the intrinsic structure of the various benchmarks, we can observe that ImpBench programs exhibit, on average, a clock-cycle count about half that of

the MiBench ones but a much smaller dispersion of values. This, once more, reveals the more diverse nature of the biomedical applications selected. We have also plotted the total number of executed instructions and μops. XTREM, which is based on SimpleScalar, implements ARM instructions through μops. We included μop statistics at this point and in the following discussion so as to better capture the workings of the underlying architecture. Overall, ImpBench programs display shorter execution times (about $\times$`0.5`) but much shorter instruction/μop counts (about $\times$`0.1`) than the MiBench ones. This agrees with the observations we made previously regarding the IPC and attests to the more control- and I/O-intensive nature of the biomedical programs compared to general multimedia programs. Last, it is interesting to observe that, although *fin* and *crc32* have smaller static code sizes than their respective counterparts *mlzo* and *checksum*, they have much larger dynamic code sizes. Given that all compression, encryption and data-integrity algorithms operate on the same input dataset (a 10-KB binary file of ECG readouts), these observed variations between static and dynamic program sizes directly expose diverse intrinsic properties of the various algorithms.

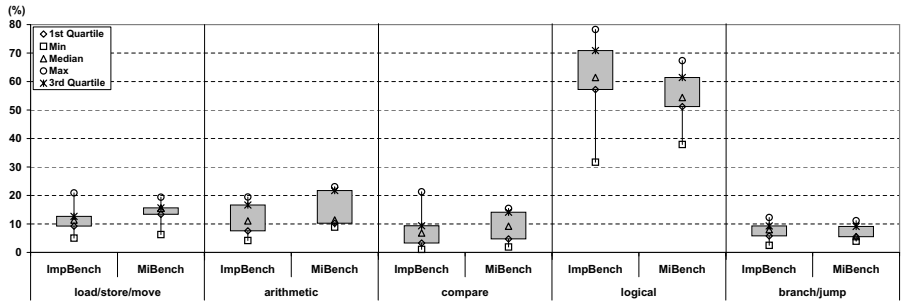
4.6.5.2 Instruction distribution

Having discussed overall instruction counts, we elaborate further on the nature of the executed instructions per benchmark suite, i.e. the instruction mix. For the same reason as before, we choose to profile μops rather than instructions. We have organized μops into five groups: data move (load, store, move), arithmetic operations (INT and FP), comparison operations, logical operations and branches (conditional and unconditional). Only the *dmu* benchmark includes floating-point operations and even that does not stress them. We wish to adhere to the initial observation that the majority of implant applications can do without or with few floating-point calculations. In effect, FP arithmetic operations are scarce or absent and have, thus, been merged with the INT operations.

Overall μop mixes are shown in Figure 4.16. We can readily observe that rates of ld/st/mov, arith, cmp and logical operations all differ significantly between the ImpBench and MiBench programs. Overall, we notice less ld/st/mov and cmp μops for ImpBench. Yet, there are more logical and br/j μops further supporting the argument that biomedical applications exhibit a more dynamic behavior than average multimedia ones. The number of arith μops is similar for both suites. We also observe that all μop categories display a larger range ($max - min$) of values for ImpBench compared to MiBench. Also, the ImpBench boxplots reveal more dispersed ld/st/mov and logical μop ratios.



(a) *Per-benchmark, average values.*



(b) *Box-and-whiskers plot.*

Figure 4.16: *Relative frequencies for load/store/move, arithmetic (int/fp), compare, logic and branch/jump instructions.*

In terms of intra-variation, *mlzo* differs largely from *fin* in all μop ratios, *misty1* differs notably from *rc6* in arith and logical μop ratios and, with the addition of cmp ratios, so does *crc32* compared with *checksum*. All in all, the variation among the various ImpBench programs is visible. A last observation to make at this point is that logical μops are clearly dominating the μop mix which is partly explained by the known tendency of the utilized ARM cross-gcc to favor the generation of logical instructions.

As previously mentioned in the analysis of compression algorithms, the XTREM simulator has been modified to collect pairs and triplets of data-dependent instructions during execution time. In Fig.4.17 we have plotted dependent-instruction combinations for all profiled benchmarks. We have limited the plot to only those combinations appearing with a frequency of `4.5%` or higher during dynamic-code execution. With this limitation, Figure 4.17 has been plotted. It reveals that ImpBench and MiBench both favor (heavily depending on the compiler used) predominantly the “and-eor” and “eor-eor”

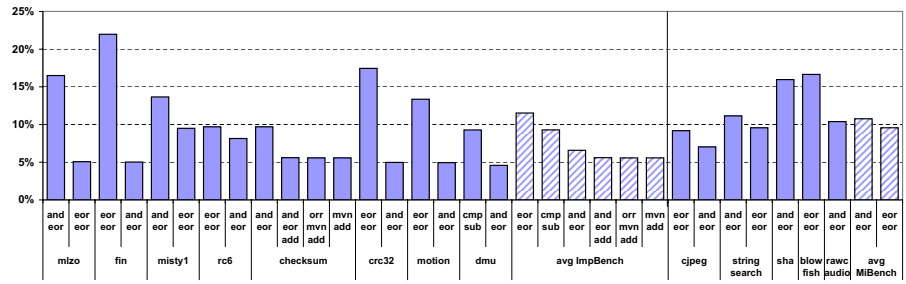


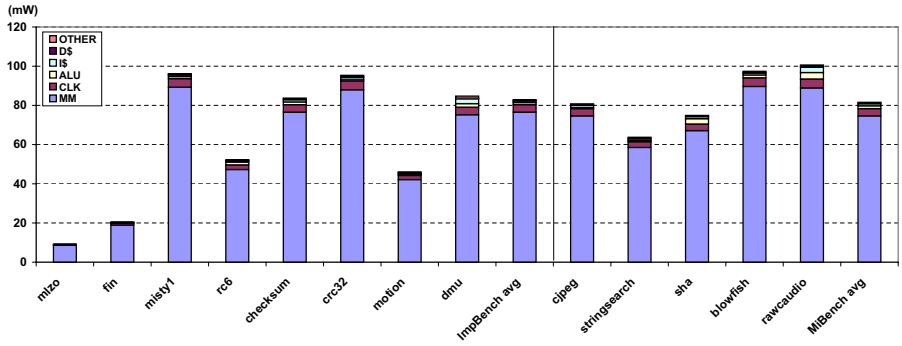
Figure 4.17: *Relative frequencies of data-dependent, dynamic-instruction combinations.*

pairs with high occurrence frequencies ("eor": exclusive-or operation). ImpBench once more illustrates a higher diversity and introduces more frequent instruction combinations. Namely, the pairs "cmp-sub" (due to *dmu*) and "mvn-add" (due to *checksum*) as well as the triplets 'and-eor-add" and "orr-mvn-add" (both due to *checksum*). As desired, it also captures different instruction dependencies within the compression, the encryption, the data-integrity and the real-programs categories.

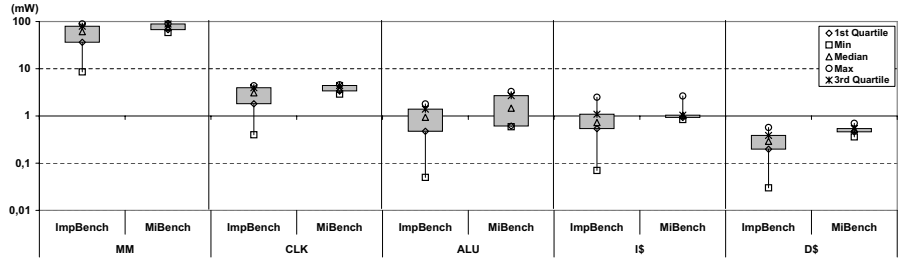
When popular instruction combinations (along with the previously discussed single-instruction frequencies) have been identified, specific microarchitectural or architectural optimizations can be made to achieve more efficient machines. For instance, in a processor design seriously limited by power and area constraints, to favor the execution of "eor-eor" or "and-eor" pairs, only data forwarding in the logical-operations circuitry of the ALU may be allowed to be incorporated. Other techniques, like the known interlock-collapsing ALUs [141], might also be considered.

4.6.5.3 Power consumption

A final metric to be evaluated against ImpBench and MiBench is average power consumption of the executed programs. This is highly relevant to embedded systems and particularly to energy-scavenging microelectronic implants. In Figure 4.18 we have plotted the per-component and overall average power consumption of our simulated processor. Overall, the average power consumption of MiBench is about $\times 1.2$ times that of ImpBench. This can be attributed partly to the fact that most ImpBench programs have been carefully picked for low-power applications [134, 135]. Yet, it also indicates that they can provide meaningful means of workload characterization for implant pro-



(a) *Per-benchmark, average values.*



(b) *Box-and-whiskers plot.*

Figure 4.18: *Per-component and overall average power consumption.*

cessors since they implicitly respect tight power budgets ever present in the considered application field.

Further analysis of the results indicates the most power-hungry unit to be the memory manager (MM) for ImpBench and MiBench programs alike, followed by the CLK, ALU, I-cache and D-cache structures. MiBench manages to stress the power consumption of the MM more than ImpBench, judging by its higher percentage in the plot. The same is true for the ALU and I-cache components. However, the ImpBench programs display, in all components except for the ALU, more data-dispersed power profiles. This finding further enforces the initial observation that biomedical programs indeed are more diverse in characteristics than general multimedia ones.

In terms of intra-benchmark variation, we can clearly see that the *compression algorithms* display by far the smallest power consumption across both suites. Last, with the exception of the two data-integrity algorithms, the two members of all other ImpBench categories vary largely between them in terms of power,

i.e. one features an average power consumption half or double that of the other.

4.6.6 Summary

In view of more structured and educated implant processors in the years to come, we have carefully put together ImpBench, a collection of benchmark programs and assorted input datasets, able to capture the intrinsic of new architectures under evaluation. In the above, we have shown that ImpBench displays considerably different characteristics than the most related MiBench. IPCs, data-cache hit rates, branch-prediction hit rates, instruction frequencies and power consumption show *increased* or sufficient variation compared to MiBench, to justify the presence of a new benchmark suite.

Besides, ImpBench is a dynamic construct and, in the future, more benchmarks will be added, subject to our ongoing research. Among others, we anticipate simple DSP applications as potential candidates as well as more "real applications" like the ones we already included.

4.7 A SiMS case study

With the simulator and all benchmarks finally in place, we are now able to put together a first, realistic application for our envisioned SiMS processor. As the closing part of this chapter, we present, next, the *case study a typical implant application* the functionality of which is implemented as executed software in the SiMS processor. Through the use of XTREM, we evaluate different application aspects using an array of metrics. With this case study, we chiefly provide a first proof-of-concept application of the implant processor and, in so doing, exhibit its potential for future implant design.

4.7.1 Implant characteristics

In order to study and simulate a representative implant application, commonly met characteristics of implantable systems need to be identified. Our prior work, has revealed a number of facts which we repeat here for convenience.

First, biomedical implants perform periodic, in-vivo measurements of physiological data through appropriate sensors. The collected data need to be stored inside the implant for later telemetry to an external monitoring/logging device. Second, data must be transmitted securely as well as reliably; information

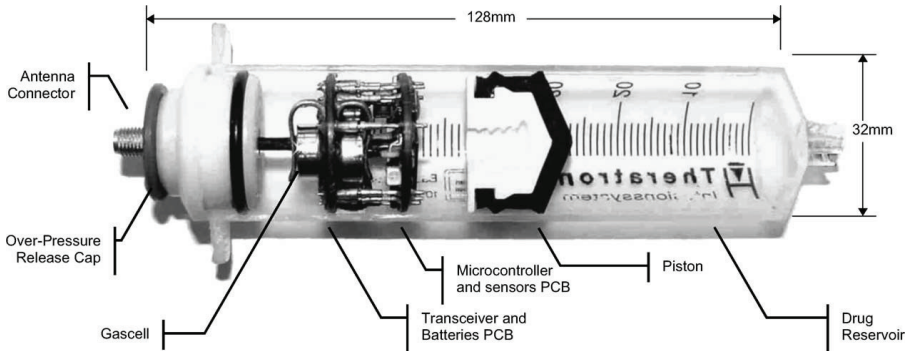


Figure 4.19: *DMU device lateral photograph (Courtesy: [25]).*

eavesdropping or loss thereof can not be tolerated. Third, open- or closed-loop control of (in-vivo) physiological parameters may be effectuated through appropriate actuators, e.g. the "artificial pancreas" application whereby insulin is released to the blood based on periodic, in-vivo, glucose-level measurements. Fourth, biological or other data manipulation in implants can in most cases be coped with through integer (INT) arithmetic. Expensive, floating-point (FP) operations can be avoided by smart manipulation of the data or postponed until the time when data is telemetered to an external logging station with infinite (in our context) computational resources. Last, typical data-memory sizes inside the implants range from `1 KB` to `10 KB`. Program memories are equally restricted, with sizes in the order of magnitude of `10 KB`.

4.7.2 Crafting a realistic application

Rather than creating an artificial and, thus, potentially biased application based on synthetic application descriptions, we chose to use a real-world scenario. As already seen in the description of the ImpBench components, Cross et al. [25] have developed intravaginal drug-delivery & monitoring units (DMUs) for regulating the oestrus cycle of dairy cows. The functionality of each DMU is implemented as embedded-C code running in a M16C, a 16-bit microcontroller (μC) from Mitsubishi. This μC is the central component in a system consisting of a transceiver module, temperature, pressure, motion and other sensors as well as a current-driven gas cell (i.e. an actuator) which is used for controlled drug release based on electrolytic-gas production (see a system photograph in Figure 4.19). According to the authors, the DMUs have been designed: i) to deliver an arbitrary and complex variable-rate profile of a viscous vehicle, ii)

to be controlled externally from the animal, and iii) to be monitored externally and provide immediate or logged data over a wireless link.

We have extracted the embedded-C code and have adapted it from the implantable system. The current program version does (and can) not simulate all real-time aspects of the actual (interrupt-driven) system, such as low-level functionality (e.g. sensor/actuator calibrations), transceiver operation and so on. Nonetheless, the emphasis here is on the computations performed by the implant core in response to external and internal events (i.e. interrupts). Having contacted the DMU designers directly, we have acquired *real data* collected from the field (e.g. temperature, pressure and current output). They have been used in our source code to drive the (simulated) run-time behavior of the actual DMU system as closely as possible.

This particular application has been selected since it incorporates all aspects we consider common and crucial in current and future implants. That is, real-time, closed-loop control of actuating elements based on sensory readouts, device self-calibration and self-check operations (e.g. battery-level check, adherence to the desired drug-delivery profile etc.), to name a few. At the same time, the application imposes low- to moderate-speed requirements on the device which, for our targeted field of ultra-low-power implants, is a desired feature. All in all, the selected application is considered highly representative for our envisioned biomedical processor.

In concordance with our discussion in Section 4.2 on future implant workloads, the basic DMU functionality has been *enhanced* with data compression, encryption and data-integrity runs. The functionality of our overall case study is illustrated as a block diagram in Fig.4.20. Over a period of approx. 10 (simulated) hours, the implant periodically (i.e. every 6 min) collects intravaginal temperature- and pressure-sensor readings and logs them. Based on those readings, it switches the gas cell on and off. This gas cell is responsible for the rate of drug delivery into the animal intravaginal space, following a user-defined drug-delivery profile. Besides, every 40 minutes, the implant performs some housekeeping tasks like safety checks and recalibrations of the sensors and actuators.

At the end of the 10 simulated hours (pure DMU operation is finished), logged data is compressed and remain stored in native memory or transmitted to an external host. Transmitted data are first compressed, then encrypted and, finally, augmented with data-integrity check bits. In order to comply with the previously described specifications, data logs of maximally `10 KB` each have been generated. All above tasks are performed in software by the implant processor.

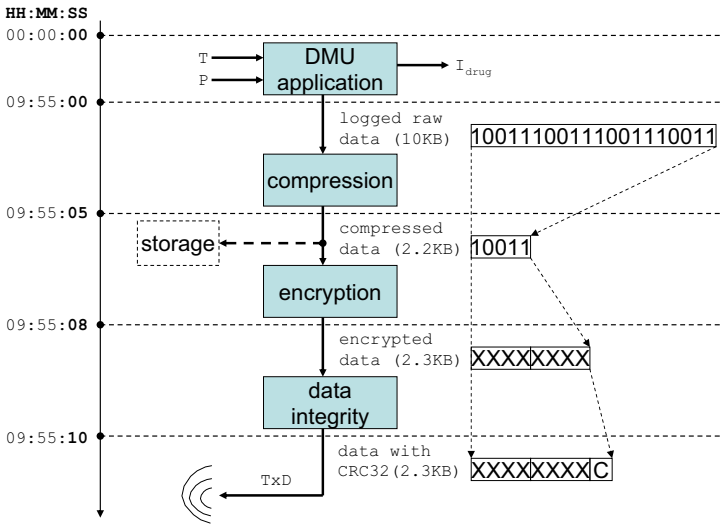


Figure 4.20: *Block diagram of simulated implant application and data-payload sizes.*

feature	value
ISA	32-bit ARMv5TE-compatible
Pipel. depth / Datap. width	7/8-stage, super-pipelined / 32-bit
RF size	16 registers
Issue policy / Instr.window	in-order / single-instruction
I-Cache, L1	64KB, 64-way assoc. (1cc hit/170cc miss)
D-Cache, L1	32KB, 2-way assoc. (1cc hit/170cc miss)
TLB / BTB	1-entry fully-assoc.
Branch Predictor	2-bit Bimodal (32-entry RAS)
Write Buffer / Fill Buffer	2-entry / 2-entry
Mem. port no / bus width	1 port / 1 Byte
INT/FP ALUs	1/1
Clock freq. / Implem. tech.	2 MHz / 0.18 μm @ 1.5 Volt

Table 4.19: *XTREM configuration for exploring a SiMS-processor case study.*

Based on our previous work, suitable algorithms in terms of performance, power, energy and size have been used for the compression (miniLZO [103]), encryption (MISTY1 [79]) and data-integrity (CRC32 [19]) operations.

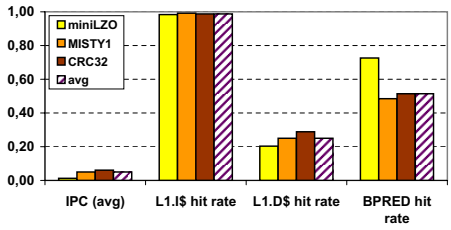


Figure 4.21: *Average IPCs, I-/D-cache hit rates and branch-prediction rates.*

4.7.3 Experimental setup

Main XTREM characteristics for this case study are summarized in Table 4.19. Various XTREM architectural parameters have been restricted or disabled to better reflect the highly constrained implant processor (also included in the table above). Concisely, the BTB has been reduced to a 2-entry, direct-mapped structure, the WB and the FB have been reduced also to 2-entry structures, the MEM width has been reduced to 1 Byte, both L2 caches have been disabled, both L1 caches have been configured based on concurrent – at the time of writing – microarchitecture optimization studies (to be discussed in the next chapter) while the number of INT/FP ALUs has been reduced to 1.

4.7.4 Profiling analysis

In order to gain insight on the behavior and requirements of the tasks executed inside the implant, various metrics have been monitored and concisely presented hereafter. As we can see from Figure 4.20, tasks are executed in a *sequential* fashion. **Execution times** are sufficiently small for this real-time application as is the case with most implantable systems. For a $10-KB$ data payload, miniLZO achieves a high compression ratio (78%) in about $5.1\ sec$. Encryption adds a small overhead in size to the compressed data due to *quantization* since MISTY1 operates on $8-Byte$ quantities. It achieves symmetric-key encryption of the compressed data in about $3.2\ sec$. Last, CRC32 data integrity adds a size overhead of $4\ Bytes$ to the payload by appending an unsigned-long-integer checksum value and costs an extra $1\ sec$ in time.

Overall, a (simulated) application execution time of $9.3\ sec$ is required to perform all data-manipulation tasks after the $10-KB$ log file has been generated; that is, an extra processing time of $9.3\ sec$ for every 10 hours of logging activity. Even though we are using a highly-resource constrained processor, the system response time is very low indicating a processor performance which

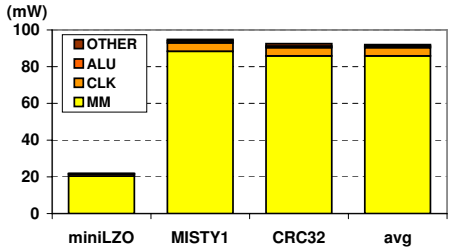


Figure 4.22: *Per-component and overall average power consumption.*

is more than adequate for the subclass of moderate-throughput applications we are targeting. To illustrate, in Figure 4.21 **Instructions Per Cycle (IPC)**, **cache-** and **branch-behavior** are depicted. The exceptionally low D-cache hits reveal strong data-locality characteristics of the biological data and hint on clear performance gains should larger D-cache sizes be allowed. Conversely, the high I-cache hits indicate that relatively small I-cache sizes[8] are sufficient due to the highly predictable program behavior of the considered tasks. Given that we have used a relatively simple branch-prediction scheme (2-bit Bimodal), BPRED rates are rather high with miniLZO scoring exceptionally high. Its IPC, though, remains the smallest due to its low D-cache hit rates. Besides, IPC is low for all programs but, as discussed previously in the execution times, it is more than sufficient for covering the real-time-application demands of the implant.

In fact, the low IPCs - as long as they cover the demands of the application - are a desired feature since they imply limited power demands on the part of the processor. This is a much sought attribute in power-starved systems as implants are. To illustrate this, overall and per-component **average power-consumption** figures for all three tasks are depicted in Figure 4.22. We can see that miniLZO consumes remarkably low power (about $20\ mW$) but, in general, all tasks consume less than $100\ mW$. The low power profile of miniLZO agrees with the lower IPC it exhibits, as previously predicted. We can further deduce from the figure that the main culprit of power consumption in the processor is the non-power-scalable memory-manager unit (MM), followed by the clock network (CLK). This indicates that the selected $2-MHz$ operating frequency is high enough for the tasks to execute in time and, at the same time, low enough to impact power consumption minimally. Besides, the known fact that MM power (in XTREM) does not scale properly with frequency does not forbid us from observing that in implantable systems as the one modeled here, the MM is

[8] See Section 5.1 in the next chapter for the cache scaling factor assumed.

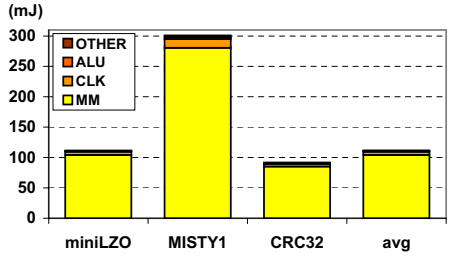


Figure 4.23: *Per-component and overall total energy expenditure.*

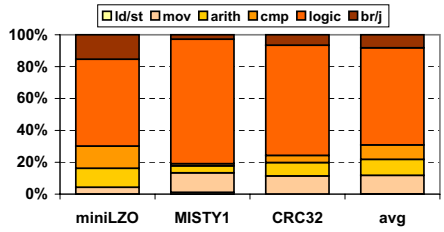


Figure 4.24: *Relative frequencies for load/store, move, arithmetic, compare, logical and branch/jump μop.*

under heavy use and should be carefully designed for low power consumption.

Except for average power consumption, it is interesting also to see what the **overall energy budgets** of the various tasks are; that is, by how much we must deplete the implant battery to perform each task. In Figure 4.23 we can see that the encryption program, MISTY1, consumes a disproportionally large amount of energy compared to the other tasks. This indicates that we should carefully select whether to encrypt the biological data or not prior to transmission, depending on the application scenario and the sensitivity of the data itself. If privacy is not required or is guaranteed through other means, e.g. transmission in a trusted environment, considerable battery reserves can be saved by disabling encryption. Alternatively, a compromise between level of provided security and consumed energy could be investigated. While this is not (currently) supported in MISTY1, future versions of it or other low-power encryption algorithms might be considered that are able to achieve such a trade-off. Overall, Figure 4.23 reveals that the energy costs of the various tasks are not necessarily identical to their power profiles and is essential in deciding which tasks can be performed at a given point in time, based on available battery-capacity levels.

The final topic of our discussion relates to the **instruction mix** of the various tasks. We, again, included μop (rather than instruction) statistics at this point

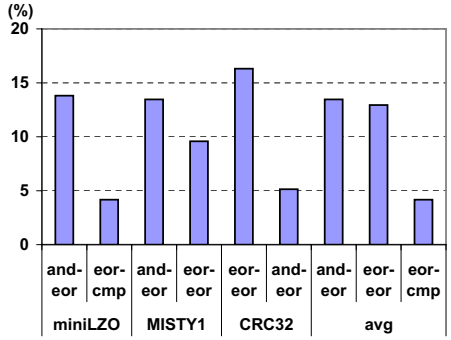


Figure 4.25: *Relative frequencies of data-dependent, dynamic-μop combinations.*

and in the following discussion so as to better capture the workings of the underlying architecture. Overall instruction mixes are shown in Figure 4.24. All programs heavily utilize *logical* μops; MISTY1 expectedly scores the highest which is typical of encryption algorithms. In terms of *arithmetic* operations, it should be stressed that all tasks (except DMU) are integer programs and miniLZO displays the highest concentration of *arithmetic* and *compare* operations. It also includes the largest ratio of *branch* or *jump* μops. MISTY1 and CRC32, on the contrary, exhibit larger ratios of data *move* μops.

In Figure 4.25, we further collect (dynamic) **data-dependent μop pairs and triplets**. μop pairs or triplets are consecutive μops whereby data generated by the first μop is consumed by the second and/or third μop; i.e. whereby data dependencies occur. We have limited the plot to only those combinations appearing with a frequency of 4% or more during dynamic-code execution. With this constraint we see that, overall, dependent "and-eor" (and: logical and) and "eor-eor" (eor: logical exclusive-or) pairs are by far the most frequent ones, followed by "eor-cmp" (cmp: compare) pairs. This observation reveals a high popularity of dependent logical-μop pairs. We, thus, get a clear indication that data-forwarding in the logical-operation part of the ALU, interlock-collapsing-ALU techniques [141] or other (micro)architectural optimizations will significantly benefit the implant processor. Further, the "eor-cmp" pair, combined with the previously seen μop mixes, gives directions on optimizing the compare-and-branch subsystem of the processor. Last but not least, all above observations on μop frequencies can give clear directions as to which instructions should be explicitly implemented in hardware and which ones can be afforded to be implemented in software (compiler-side conversion).

4.7.5 Discussion

To sum up our findings, based on the selected biomedical application, we can support the viability of a highly resource-constrained, novel processor for implants. Featuring a low as $2-MHz$ clock frequency and small I/D-caches, it will be able to meet its real-time goals for a broad range of application scenarios similar to or simpler than the one described here. Furthermore, it will feature a low average power profile of less than $100\ mW$, and - excluding data encryption - a similarly low energy profile of less than $300\ J$ (overall) per executed task.

It should be noted, however, that at the area and power penalty of a slightly increased D-cache size, program execution times will drop significantly as the simulations indicate. This will, in turn, lead to an even lower energy profile. The next chapter will provide a more extensive analysis of the energy benefits incurred with a bigger D-cache. Besides, reported power/energy figures are likely to be higher than actual ones since the XTREM simulator was not aimed at the ULP application spectrum. What is more, compression and encryption algorithms designed for or tuned to implantable systems, should assist further in this direction.

Last, explicit microarchitectural optimizations of the envisioned processor will drive power and energy figures further down. Hardware provisions for favorable execution of *logical* and, secondarily, *arithmetic/compare* μops as well as of specific *logical/compare* μop pairs must be incorporated in the design.

4.8 Summary

This chapter has laid the seminal work for the design of the SiMS processor. A simulator has been selected and its suitability for the purpose intended has been demonstrated, albeit in the presence of bugs. Through use of this simulator, a large-scale exploration and analysis of suitable workloads for future implant processors has been conducted. This analysis, apart from highlighting optimal candidates across various metrics such as performance, power, energy and memory footprint, has also offered ample hints towards the optimal design of the SiMS processor. To the best of our knowledge, this is the first such analysis and has led to the creation of ImpBench, a novel benchmark suite that aspires to be a reference platform for designing and comparing implant processors. Last, in this chapter we have presented a case study of the first, complete and realistic application to be run on the envisioned SiMS processor.

Note. The content of this chapter is partly based on collaborative work with D. Zhu, ir., and C. Kachris, dr., and has resulted in the following papers:

C. Strydis, G. N. Gaydadjiev, **Profiling of Lossless-Compression Algorithms for a Novel Biomedical-Implant Architecture**, 6th IEEE/ACM/IFIP international conference on Hardware/Software codesign and system synthesis (CODES'08), pp. 109-114, Atlanta, Georgia, USA, October 2008.

C. Strydis, D. Zhu, G. N. Gaydadjiev, **Profiling of Symmetric-Encryption Algorithms for a Novel Biomedical-Implant Architecture**, ACM International Conference on Computing Frontiers (CF'08), pp. 231-240, Ischia, Italy, May 2008.

C. Strydis, C. Kachris, G. N. Gaydadjiev, **ImpBench: A Novel Benchmark Suite for Biomedical, Microelectronic Implants**, IEEE International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (IC-SAMOS VIII), pp. 86-95, Samos, Greece, July 2008.

C. Strydis, G. N. Gaydadjiev, **The Case for a Generic Implant Processor**, 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC'08), pp. 3186-3191, Vancouver, Canada, August 2008.

5

SiMS-processor microarchitecture evaluation

EXTENSIVE work has been done, as detailed in the previous chapter, in identifying and profiling common workloads to be executed on the targeted implant architecture. Algorithms for lossless data compression, symmetric-key encryption and data integrity as well as representative real-world applications have been evaluated and suitable candidates have been isolated. Moreover, a carefully selected benchmark suite for microelectronic implants (ImpBench) has been proposed, based on the profiled applications, to guide and assist future implant design. This benchmark suite has been shown to offer diverse program behaviors and, thus, to be able to capture corners of our design space.

In this chapter, we rely on our established simulation environment to offer an in-depth exploratory study on suitable cache organizations and branch-prediction policies for our envisioned SiMS processor. Standard, first-order, optimization goals *performance*, *power consumption* and *energy expenditure*, that we have employed so far, are in this chapter expanded by a third one, *area utilization*.

5.1 Evaluation of cache organizations

We profile various instruction- and data-cache organization alternatives for the SiMS processor against metrics of performance, power, energy and area. We, then, select the ones with the best characteristics for the targeted application domain. We, thus, offer insights on the design and implementation of the cache subsystem of the targeted processor. Concisely, the contributions of this evaluation are as follows:

- With respect to a given collection of typical and representative biomedical workloads, to specify optimal I- and D-cache geometries under performance, power, energy and area constraints;
- To offer original, quantitative data on the behavior and specifications of I- and D-caches for current and future implant processors; and
- To propose a sound methodology and toolset for selecting optimal I- and D-cache geometries for different biomedical (or other) workloads.

A problem with related evaluation works is that caches are studied in isolation from the rest of the system and, thus, no overall performance behavior is attached to the various power figures, while information about the interplay between different cache configurations and other components of a processor core cannot be acquired. The work presented here is original in that it targets a different class of low-power devices with particular idiosyncrasies. To the best of our knowledge, no similar effort has been reported so far in explicitly studying cache structures for an implant processor. Furthermore, we have considered aspects of performance and power but also energy and area in our study, to drive our selection process.

5.1.1 Experimental setup

In order to correctly set up our experiments as well as to select suitable cache geometries, to be discussed in the following section, we first elaborate on the particular idiosyncrasies of microelectronic implants. Such implants are highly resource-constrained devices. The (re)implantation frequency for battery replacement - a costly and risky undertaking - is directly related to the operational life of a device. In order to achieve long in-vivo operation times, we are aiming at a **tight power budget** (μW order of magnitude).

An **ultra-small form factor** is also required for such devices considering the space available for implantation inside the body. This means that available processor area is also limited. Besides there are further aspects benefitting from low transistor counts (but out of the scope of this work) such as higher device yield, increased testability and higher coverage for fault-tolerant design.

There has been shown to exist in Chapter 2 and we are targeting a significant category of biomedical applications displaying **moderate performance** requirements, e.g. a feedback loop periodically regulating the functionality of bioactuators based on readouts from biosensors. Even so, under tight power

Benchmark	name	size (KB)
Compression	miniLZO [103]	16.3
	Finnish [29]	10.4
Encryption	MISTY1 [79]	18.8
	RC6 [79]	11.4
Data integrity	checksum [16]	9.4
	CRC32 [19]	9.3
Real applications	motion [152]	9.44
	DMU [25]	19.5

Table 5.1: *ImpBench components and (static) binary size.*

and area budgets, the implant still has to complete its real-time (repetitive) duties within specific time margins. To do so, it must maintain a minimal instruction rate under the worst-case scenario.

5.1.1.1 Input datasets

For the work presented in this cache-evaluation study, we have used the $10-KB$ ECG dataset representative of all workloads in our disposal (see Table 4.2). Our implant survey has revealed that typical memory sizes inside the implants range from $1\ KB$ to $10\ KB$; thus, the use of $10-KB$ ECG data.

5.1.1.2 Benchmarks

All eight benchmarks of the ImpBench suite have been used, namely the *lossless data compression* algorithms, the *symmetric-key encryption* algorithms, the *data-integrity* algorithms and the *real applications*. The benchmarks are reported once more in Table 5.1 for convenience; binary sizes are also included to give an idea about their code complexity.

5.1.1.3 Simulation testbed

Our cache evaluation study has been based on the XTREM **simulator**. As previously mentioned in Section 4.1, XTREM allows monitoring of 14 different functional units of the Intel XScale core: Instruction Decoder (DEC), Branch-Target Buffer (BTB), Fill Buffer (FB), Write Buffer (WB), Pend Buffer (PB), Register File (REG), Instruction Cache (I$), Data Cache (D$), Arithmetic-Logic Unit (ALU), Shift Unit (SHF), Multiplier Accumulator (MAC), Internal Memory Bus (MEM), Memory Manager (MM) and Clock (CLK). However,

feature	value
ISA	32-bit ARMv5TE-compatible
Pipeline depth	7/8-stage, super-pipelined
Datapath width	32-bit
RF size	16 registers
Issue policy/Instr.window	in-order/single-instruction
I-Cache, L1	**2B/block, 1-cc hit/170-cc miss lat.**
D-Cache, L1	**2B/block, 1-cc hit/170-cc miss lat.**
BTB/TLB	2-entry direct-mapped/1-entry
Branch Predictor	2-bit Bimodal (32-entry RAS)
Write Buffer	2-entry
Fill Buffer	2-entry
Mem. bus width	1 Byte (1 mem. port)
INT/FP ALUs	1/1
Clock freq.	2 MHz
Implem. tech.	0.18 μm @ 1.5 Volt

Table 5.2: *XTREM configuration for study on cache geometries.*

to better match our application field and, also, to isolate cache behavior as much as possible, many of XTREM's architectural parameters have been cut down or disabled to better reflect the highly constrained implantable processors. The modified XTREM characteristics are summarized in Table 5.2. As has been highlighted in bold in the table, in this study we are interested in the optimal configuration for the two L1 cache structures (i.e. the instruction- and data-cache units).

Last, in order to be able to evaluate a large number of cache configurations and automatically gather, plot and evaluate the findings, we have developed software wrappers in Perl. Through these wrappers, we were able to run simulations on thousands of different cache configurations and aggregate the monitored metrics into few, manageable figures. The wrappers were also responsible for boundary control of the tested geometries and error checking of the XTREM simulator. Illegal or erroneous simulator stated have been pruned out and excluded from the conclusions of this study. Moreover, the CACTI tool which has been used for cache-size estimations (to be discussed later on), has been interfaced to the simulator to automatically generate and return figures for different cache sizes. This Perl-based simulator ensemble has allowed us to perform a large number of runs with diverse cache and other simulator parameters and could easily be used in a score of other profiling studies with little to moderate modifications.

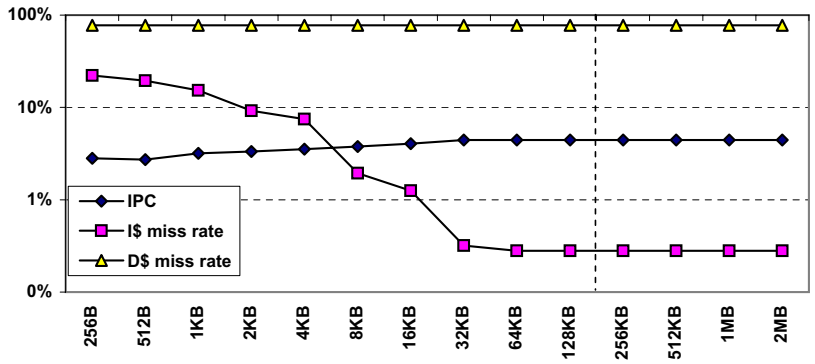


Figure 5.1: *Averaged, average IPC and I/D-cache miss rates for various direct-mapped, I-cache sizes.*

5.1.2 Profiling analysis

The XScale core (and thus XTREM) assumes a Harvard architecture, with separate L1 I-cache and D-cache and no L2 caches so as to relax the bandwidth requirements on the memory bus. Most implantable systems we have studied so far feature separate caches (or memories, in general), and thus we have limited our study to split caches as well.

To perform a thorough but realistic investigation of cache sizes for biomedical implants, we have evaluated sizes in the range: [32B, 16KB], in accordance with our prior study of existing implantable devices. However, as seen in Table 5.2, XTREM simulates a 32-bit wide architecture which is unrealistic for the ultra-low-power processor that we are targeting. By conservatively assuming an average size of 8-bits for our implant-processor ISA, we had to scale up by a factor of $\times 4$, to move from 8-bit to 32-bit quantities (which are supported by our simulation testbed). Further, since the minimal block size supported by XTREM is 2, the scaling factor becomes $\times 8$. In effect, the properly scaled, final cache-size range becomes: [256B, 128KB]. We are well aware that the final, scaled-down findings might be suboptimal when mitigated to our actual implant processor however they will give us useful hints and a good starting point for further architectural design-space exploration. All 8 benchmarks have been profiled against each cache size and average values are reported.

5.1.2.1 Cache sizes

The first step in our methodology involves finding the pair of optimal L1 I- and D-cache sizes under constraints of performance, power, energy and area. First, we have kept D-cache size constant at 32 *KB* and we have gradually increased I-cache size from 256 *B* to 128 *KB*, each step featuring double the size of the previous one. Both caches have been configured as direct-mapped structures for this step. Figure 5.1 illustrates the variation of IPC and I/D-cache miss ratios as a function of I-cache size. The figure actually plots also larger cache sizes to give a better overview of the observed trends, but such excess sizes are not considered as viable for our application domain.

Expectedly, the D-cache miss rate is not affected by the I-cache size, while the I-cache miss rate drops rapidly and practically assumes a constant miss rate at 32 *KB* and onwards. This affects the IPC which assumes a constant value at the same point. This comes as little surprise since each benchmark in our collection (see Table 5.1) essentially fits in the I-cache for sizes of 32 *KB* or more. However, it is interesting to observe that the IPC value does not, in an overall, change drastically with improving miss rates (viz. from 0.027% it saturates at 0.044%).

The next metric we examine is average power consumption. In Figure 5.2, total and per-component power figures are plotted for the investigated I-cache sizes. XTREM components with zero power consumption have been omitted from the plot. We can readily see that, while the I-cache power increases exponentially with size, it is one to two orders of magnitude smaller than that of the main power culprit: the MM unit[1]. The decoder and ALU components present the most notable increase in their power profile with increasing I-cache size, in response to the increased IPC, while the clock, D-cache and memory bus in fact display dropping power trends. Overall, average power consumption in the processor reaches a minimum for an I-cache size of 64 *KB*.

Apart from average power consumption, for embedded systems with a very constrained energy budget such as implants are, it is also important to examine the overall energy spent by the processor for executing all assigned tasks. Energy has been shown to depend heavily on execution time and, thus, energy plots are not necessarily identical to power plots. In Figure 5.3 overall energy budgets for different I-cache sizes are plotted. Energy profiles in this case are similar to the power profiles with the minimum again observed for the $64-KB$

[1] Remember that, as discussed in Chapter 4, the MM power does not scale properly with frequency – contrary to the other XTREM subsystems.

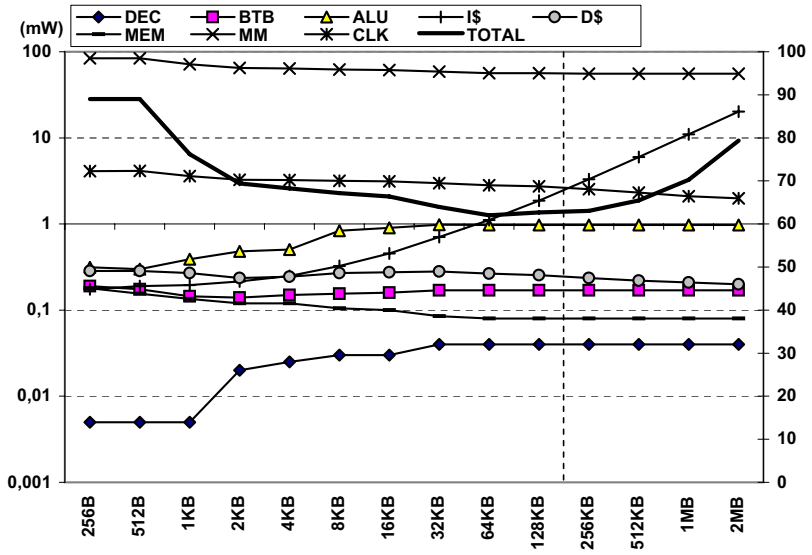


Figure 5.2: *Averaged, total (right axis) and per-component (left axis) average power consumption (in mW) for various direct-mapped, I-cache sizes.*

case. However, as can be observed from the "TOTAL" line plots, contrary to power, energy budget drops more steeply in the range from $256\ B$ to $64\ KB$ which makes moving to I-cache sizes smaller than $64\ KB$ more attractive.

In a fashion identical to I-cache sizes, we further investigate D-cache sizes. In Figure 5.4, the average IPC and I/D-cache miss rates for a constant I-cache size of $32\ KB$ and variable D-cache sizes are plotted. Contrary to I-cache behavior, we can readily observe that increasing the D-cache size has minimal impact on its miss rate. To be precise, D-cache miss rates drop from an initial maximum of 0.863% to a final minimum of 0.776% (first observed at $512\ KB$) for our selected benchmark suite. I-cache miss rates by comparison present a *proportionally* larger drop in the locus of $64\ KB$, but in absolute terms remain roughly unaffected by the D-cache size. In effect, the IPC presents a positive peak at $1\ KB$ but then stabilizes to a constant value for a $64 - KB$ size and onwards.

As far as power consumption is concerned, results are plotted in Figure 5.5. With the exception of the clock network and of course the D-cache itself, D-cache size increases do not affect other processor subsystems as radically as the I-cache. The IPC spike observed in the previous figure, manifests here also

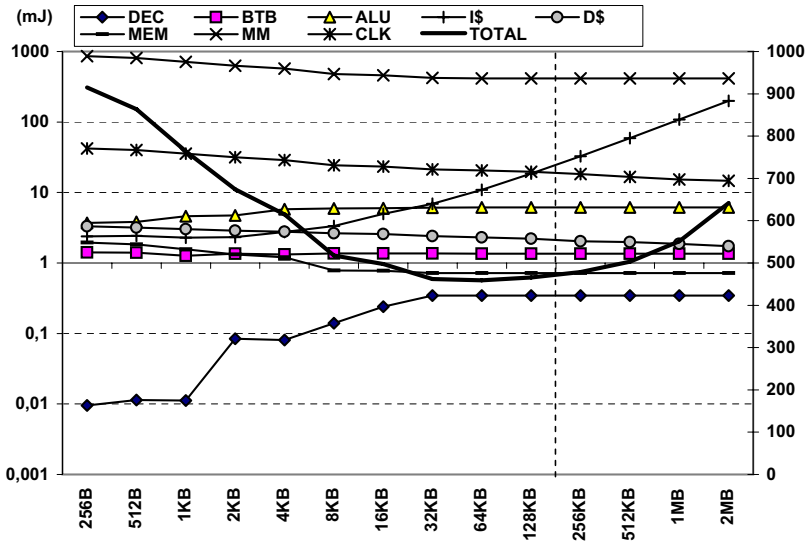


Figure 5.3: *Averaged, total (right axis) and per-component (left axis) energy budget (in mJ) for various direct-mapped, I-cache sizes.*

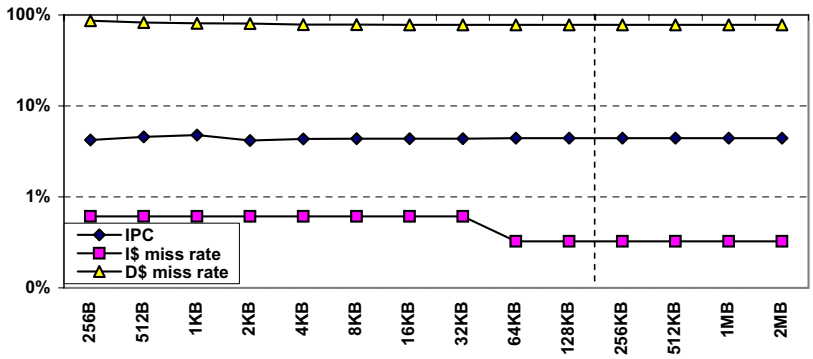


Figure 5.4: *Averaged, average IPC and I/D-cache miss rates for various direct-mapped, D-cache sizes.*

as a power spike in the locus of $1\ KB$. Minimum power is located again in the $64 - KB$ locus but, opposite to the I-cache case, overall power consumption drops steeply to this value immediately after D-cache sizes of 2 to $4\ KB$. A last observation is that, in an overall, I-cache size variation has a stronger impact on power than D-cache size variation.

Figure 5.6, last, illustrates energy results for various D-cache sizes. As was

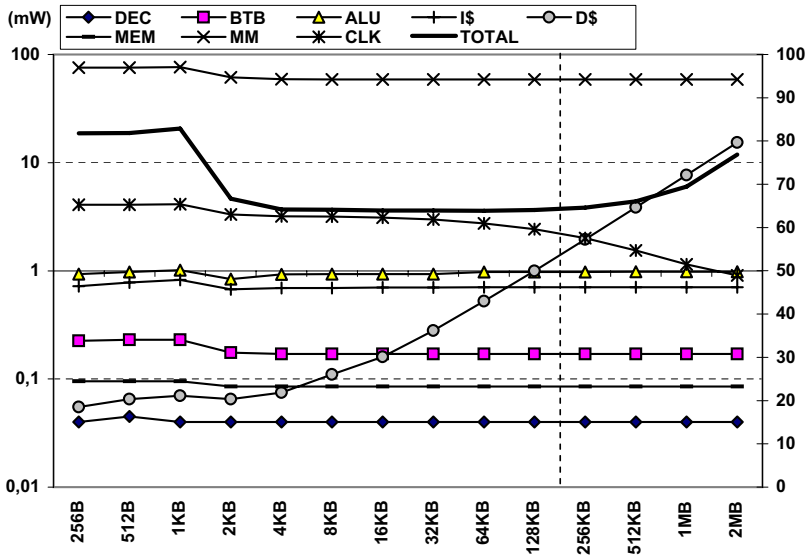


Figure 5.5: *Averaged, total (right axis) and per-component (left axis) average power consumption (in mW) for various direct-mapped, D-cache sizes.*

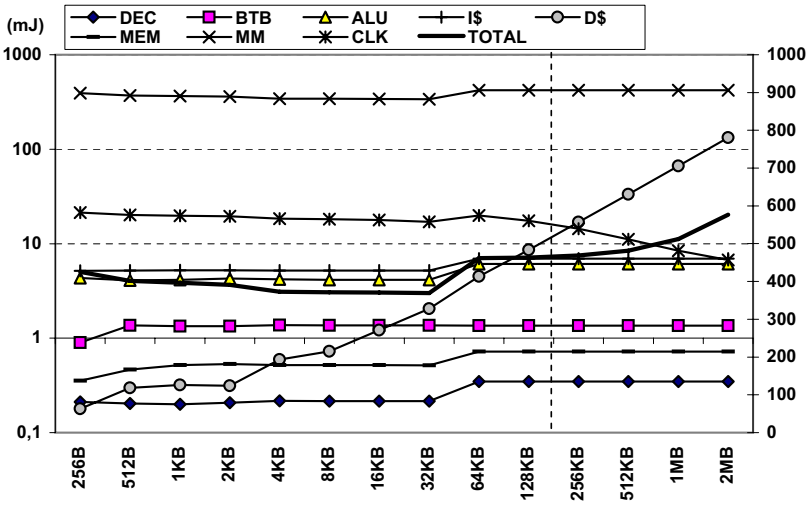


Figure 5.6: *Averaged, total (right axis) and per-component (left axis) energy budget (in mJ) for various direct-mapped, D-cache sizes.*

also the case with power, D-cache size variation has a smaller impact on energy than I-cache variation. However, the energy and power profiles in the D-cache case are less consistent between them. Minimum energy in this case is clearly observed in the $32 - KB$ locus, followed by a steep ramp-up for larger sizes. This gives us a clear indication that, energy-wise, we should focus on D-cache sizes of $32\ KB$ or less.

For selecting the best sizes for the I-cache and D-cache structures, we have based our evaluation on performance, power-consumption and energy-expenditure figures. As a performance metric, we have chosen the IPC instead of the cache miss rates since we do not wish to study the caches in an isolated environment but, rather, to capture overall system performance as a function of cache size. That is why we have used a processor (rather than cache) simulator as our testbed. For the very same reason we have also used total average power consumption and total energy budget as our second and third metric, respectively.

To find optimal solutions, we have used the following formula as our **objective function** for *minimization*:

$$F(x) = IPC_{PD}(x) + P_{PD}(x) + E_{PD}(x), \tag{5.1}$$

where x represents a single cache-size node. Each $VAR_{PD}(x)$ term in formula (5.1) represents the **percentage difference** between the VAR value at node x and the best VAR value (maximal value for IPC, minimal value for power and energy) across all cache-size nodes. This percentage difference is, in turn, given by the formula:

$$VAR_{PD}(x) = \frac{|VAR(x) - VAR_{OPT}|}{(1/2) * (VAR(x) + VAR_{OPT})} * 100, \tag{5.2}$$

where $VAR_{OPT} = max(VAR(x))$ or $min(VAR(x))$, with x in the range [256B, 128KB]. We have chosen to use percentage differences in our objective function (5.1) so as to normalize all involved variables by calculating their "relative" deviation from the per-case optimal value.

We initially sought a cache size that optimizes all three imposed metrics. For the case of the I-cache, the size of $64\ KB$ (or $8\ KB$ for our targeted implant processor) gave the best results across IPC, power and energy. This is reflected in Table 5.3 which shows precisions levels of 1.000 for all metrics; in essence, no compromises had to be made in the decision.

Some commenting on this result is needed here. It is obvious that we have avoided including an area metric in the objective function above. The reason

	PRECISION LEVELS	
metric	**I$-size:** 64 *KB*	**D$-size:** 32 *KB*
IPC	**1.0000**	**0.9650**
power	**1.0000**	**0.9700**
energy	**1.0000**	**0.9700**

Table 5.3: *Precision levels [0.000: worst, 1.000: best] for IPC, power and energy in I/D-cache-size objective functions.*

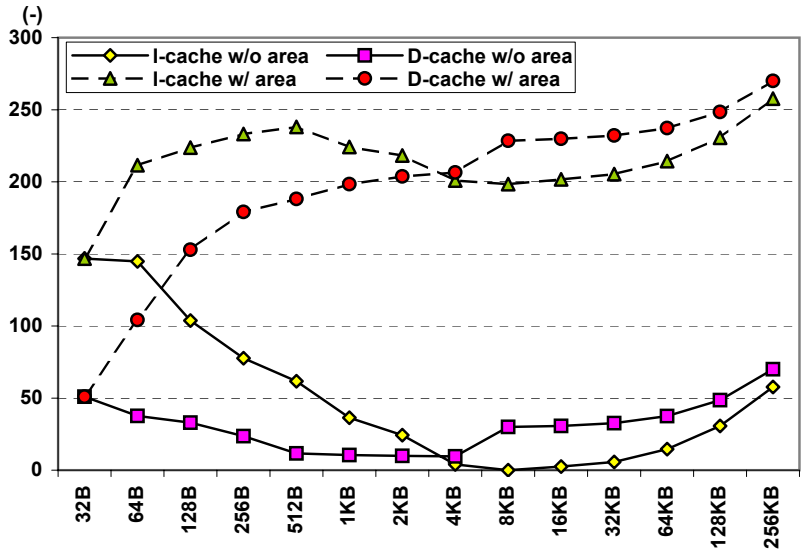


Figure 5.7: *Results for various I- and D-cache sizes of objective function* (5.1).

for this is the following: Area doubles with each increasing node and this represents a large percentage difference resulting in the "optimal" value for the area to be the very first node (smallest size). Further, due to this doubling of values, the area metric becomes dominant compared to the other three metrics which are changing slowly by comparison. In effect, the objective function would be strongly dominated by the area metric, returning the smallest cache size as the optimal one. At this point, we do not have a specific upper bound for the overall (and, thus, also cache) size of our targeted processor nor can we make any educated guess about the weight of the area metric in the objective function. We, thus, chose to omit the area metric and conclude that, under no area constraints, for the given representative benchmark collection, the optimal I-cache size is 8 *KB* for our processor.

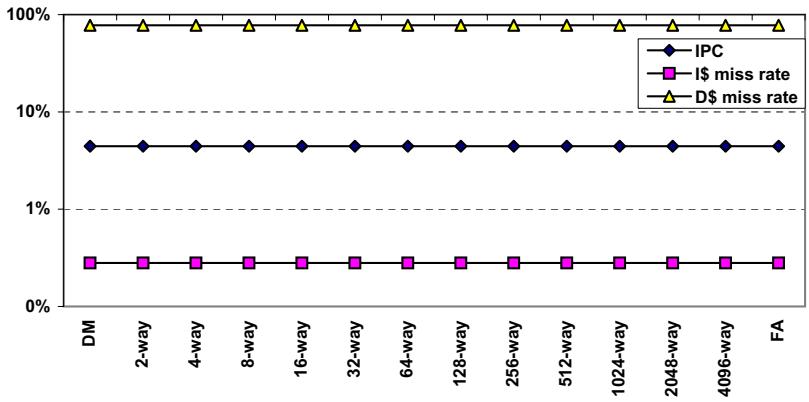


Figure 5.8: *Averaged, average IPC and I/D-cache miss rates for various I-cache associativity degrees.*

However, for the D-cache case optimal results are not directly related to program size and, what is more, are more dispersed, as has been also observed in the previous discussion on power and energy profiles. In this case, we had to lower the objective-function precision levels down to the point that we found a valid D-cache configuration. As can be seen from Table 5.3, a slight *bias* has been given to power and energy over IPC for two reasons: i) the IPC displayed insignificant variations with increasing D-cache sizes, and ii) in our targeted processor we consciously want to emphasize more on achieving low power and energy and less on performance. The D-cache size giving the best results across all three metrics was 32 *KB* (or 4 *KB* for our implant processor). For the same reasons as for the I-cache case, the area metric has been omitted here, too. Cumulative results for objective function (5.1) for various *direct-mapped* I- and D-cache sizes are given in Figure 5.7 where the effect the area metric would have - should it be included - is also shown.

5.1.2.2 Cache associativity

Having selected optimal I- and D-cache sizes, we affix our simulator I/D-caches to 64 *KB* and 32 *KB* respectively and move to the second step of our study, which is the evaluation of different degrees of associativity for both structures. Starting with the I-cache, in Figure 5.8, IPC and miss-rate results are plotted for various associativity configurations, ranging from direct-mapped (DM) to fully associative (FA). It can be easily observed that increas-

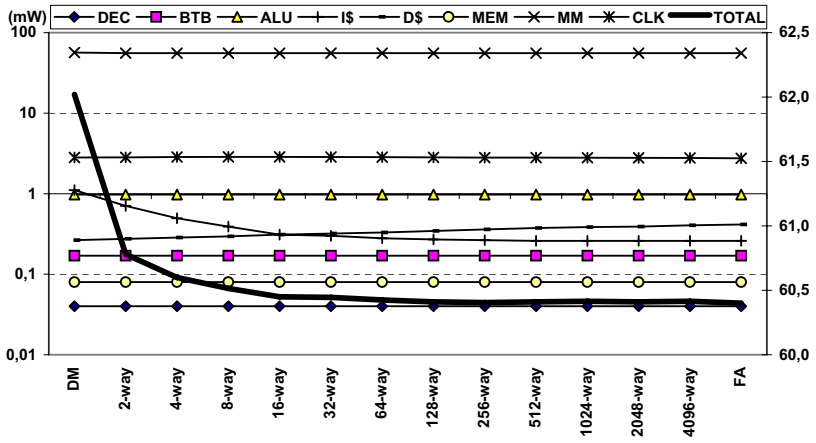


Figure 5.9: *Averaged, total (right axis) and per-component (left axis) average power consumption (in mW) for various I-cache associativity degrees.*

ing the I-cache ways has no effect on the processor performance. Going back to Figure 5.1, we can recall that with an I-cache of 64 *KB* (and onwards) the miss rate was essentially eliminated. In effect, the IPC figure here points towards a direct-mapped or few-way organization for the I-cache.

In Figure 5.9, power figures are given for various I-cache ways. As expected, changing the cache associativity hardly affects the power behavior of the processor subsystems except, of course, for the I-cache itself. It is interesting to see that although required hardware area increases with the ways, overall I-cache power consumption drops. We attribute this to the way the cache is constructed (e.g. cache-line buffering etc.). In a processor employing aggressive low-power techniques such as XScale (and, thus, XTREM) is, increasing the number of ways implies reducing the number of active sets per cache access and, thus, the cache overall power consumption. On the other hand, from the same figure we can also observe a slight increase in the D-cache power when more ways in the I-cache are implemented. Given that the IPC is not notably impacted, we have so far been unable to find the reason for that phenomenon. In any case, the combined result of the above two cache trends (plus an initial drop in the MM unit) is a net decrease of the overall, average power consumption in the processor which, for the considered ultra-low-power implants we envision, is non-negligible. In effect, with 64 ways or more the I-cache power consumption settles to its overall minimum.

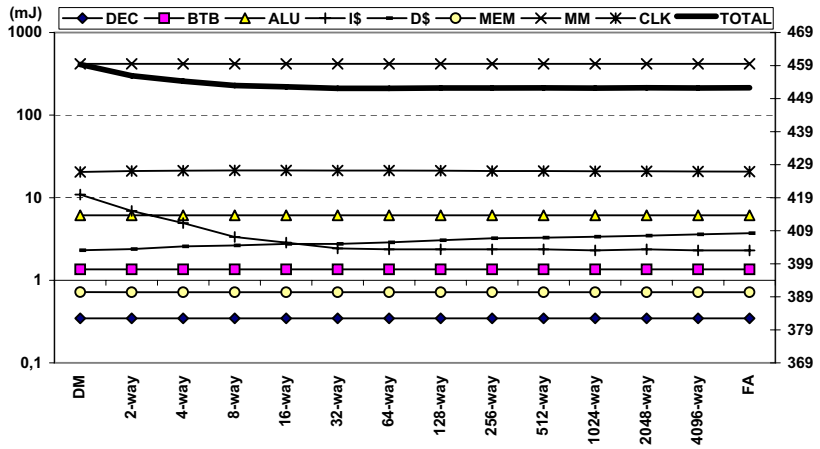


Figure 5.10: *Averaged, total (right axis) and per-component (left axis) energy budget (in mJ) for various direct-mapped, I-cache associativity degrees.*

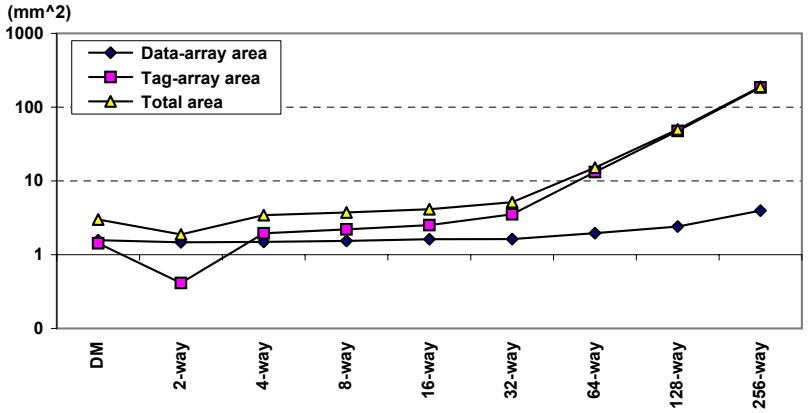


Figure 5.11: *Data-array, tag-array and total area (in $\mathtt{mm}^2$) for various I-cache associativity degrees.*

The energy budgets for different I-cache ways are illustrated in Figure 5.10. In a similar manner to power, albeit slower, overall energy costs drop with more cache ways due to the previously discussed I-cache and D-cache behaviors. At the 32- to 64-way nodes, the processor achieves the lowest energy expenditure throughout.

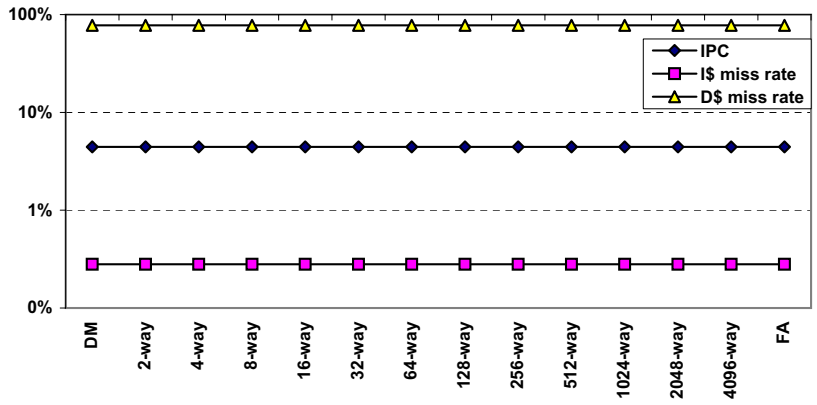


Figure 5.12: *Averaged, average IPC and I/D-cache miss rates for various D-cache associativity degrees.*

In this part of our analysis, it makes sense to also consider the *area cost* of the I-cache when moving to more associativity ways. Since moving to a higher associativity degree while keeping the overall cache size constant results in a slower area increase compared to doubling the cache size (with a given associativity degree), our objective function shall only show *weak biasing* towards the area metric, in this case. We have, therefore, properly configured and run CACTI v6.0 to collect area-utilization figures for various cache geometries. Findings for up to a realistic number of ways are illustrated in Figure 5.11. We can easily observe that the global area minimum lies at an associativity degree of 2. The 4-way or 8-way configurations are also attractive alternatives with similar area costs.

We now move to investigating the optimal D-cache associativity. Figure 5.12 reveals that changing the associativity of the D-cache has the same marginal effect to the IPC as for I-cache. Miss rates are equally unaffected, the reason being that higher associativity does not seem to offer any *additional* speedup to the execution of the benchmarks.

As far as power consumption is concerned, Figure 5.13 has been plotted. As expected, I-cache power does not change significantly with D-cache associativity, while the power consumption of the D-cache gradually drops. Overall power presents a slowly rising trend mainly due to the contributions of the clock network and the MM unit. This implies that, in the general case, less ways for the D-cache should be sought in terms of power, but the correlation is weak.

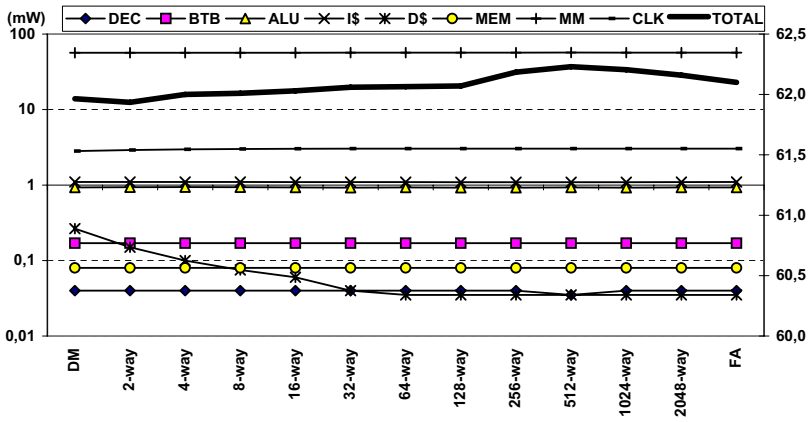


Figure 5.13: *Averaged, total (right axis) and per-component (left axis) average power consumption (in mW) for various D-cache associativity degrees.*

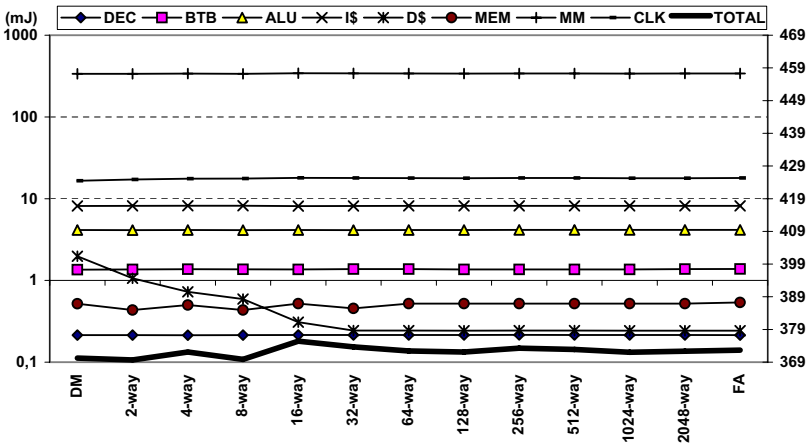


Figure 5.14: *Averaged, total (right axis) and per-component (left axis) energy budget (in mJ) for various D-cache associativity degrees.*

D-cache associativity versus energy cost has been plotted in Figure 5.14. Observations are identical to the ones previously made for power, however in this case we witness a less uniform profile in the memory bus, the MM unit and other components, resulting in a high-energy spike at the 16-way node. This prepossesses us in favor of a D-cache design with less than 16 ways.

Last, the D-cache area cost with increasing associativity has been plotted in Fi-

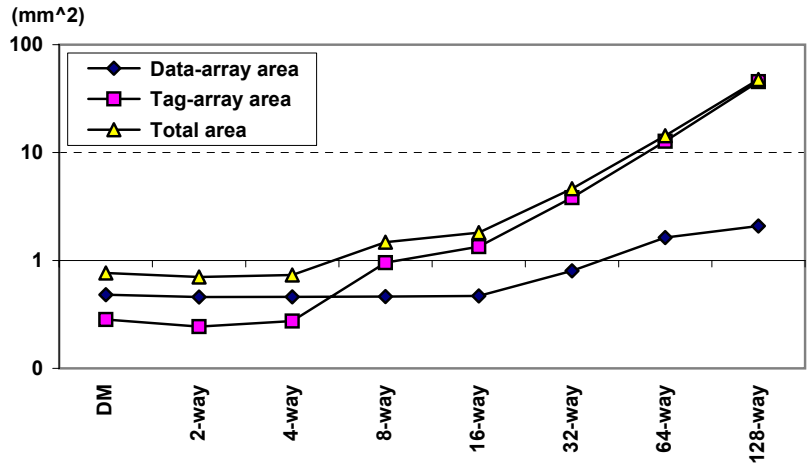


Figure 5.15: *Data-array, tag-array and total area (in mm^2) for various D-cache associativity degrees.*

PRECISION LEVELS		
metric	**I$-assoc.: 2-way**	**D$-assoc.: 2-way**
IPC	**0.9985**	**0.9985**
power	**1.0000**	**1.0000**
energy	**1.0000**	**1.0000**
area	**0.9865**	**0.9900**

Table 5.4: *Precision levels [0.000: worst, 1.000: best] for IPC, power, energy and area in I/D-cache-way objective functions.*

gure 5.15. As was the case for the I-cache, again the globally minimal area cost is found for 2-way associativity with the direct-mapped and 4-way alternatives being also viable choices.

For identifying the best I-/D-cache associativity degrees, we have used a cache-associativity objective function similar to (5.1) and percentage differences given by (5.2). The new objective function (5.3) varies only in the fact that the area percentage difference has been incorporated in the sum:

$$F(x) = IPC_{PD}(x) + P_{PD}(x) + E_{PD}(x) + A_{PD}(x), \tag{5.3}$$

We have once more (see Table 5.4) favored power and energy slightly more than performance and, in this case, area. In so doing, we have acquired the best associativity degree for both the I-cache and for the D-cache to be 2-way.

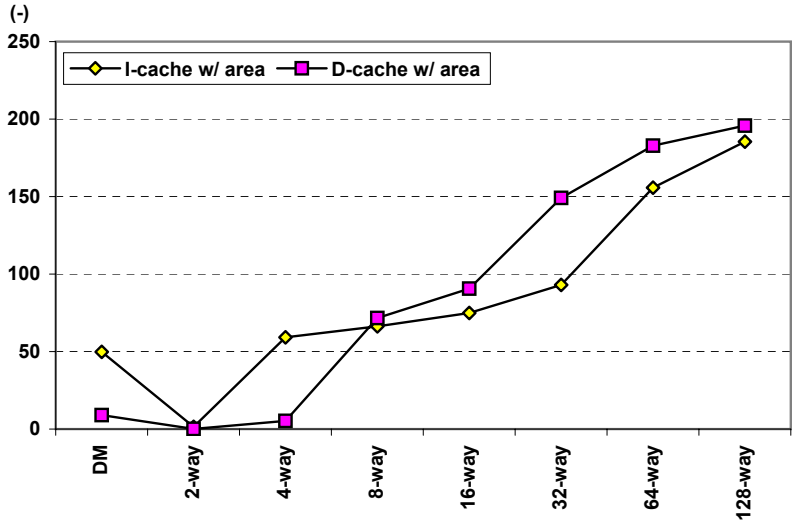


Figure 5.16: *Results for various I- and D-cache associativity degrees of objective function* (5.3)*.*

For convenience, cumulative results for objective function (5.3) for various I- and D-cache associativity degrees and fixed sizes are given in Figure 5.16.

5.1.3 Conclusions

In this evaluation study we have provided a detailed investigation of various instruction- and data-cache configurations. We have run sequential optimization (here: minimization) functions on the specified design space and have identified best instruction- and data-cache candidates for our end goal which is the design of an implant processor. Concisely, a 2-way L1 instruction-cache of 8 *KB* size and a 2-way associative L1 data-cache of 4 *KB* size have been selected. We are fully aware of the fact that we have scaled our simulation parameters and the produced results to reflect the targeted biomedical-implant processor. This does not necessarily mean that these results will represent optimal selections in our final processor design but, rather, an educated starting point for our design-space exploration.

BENCHMARK TYPE	NAME	SIZE (KB)	C.CYCLES (#)	INSTR. (#)	BRANCHES (#)	BR.RATIO (%)
Compression	miniLZO	16.30	32,482,046	199,163	34,008	17.08
	Finnish	10.40	80,581,690	852,663	147,971	17.35
Encryption	MISTY1	18.80	41,006,520	1,268,465	63,086	4.97
	RC6	11.40	25,919,634	864,930	60,869	7.04
Data integrity	checksum	9.40	1,102,562	62,869	7,933	12.62
	CRC32	9.30	12,021,257	419,159	69,976	16.69
Real apps.	motion	9.44	25,891,030	859,371	130,773	15.22
	DMU	19.50	483,432,846	36,808,268	4,393,796	11.94

Table 5.5: *ImpBench components and useful general statistics.*

5.2 Evaluation of branch-prediction schemes

By building upon the previous study on suitable cache geometries for the SiMS processor, in this section we investigate different branch-prediction alternatives under varying L1 I- and D-cache configurations. The approach we take in analyzing the various schemes is identical to the one we took for the cache study and it makes the following contributions:

- Careful evaluation of various branch-prediction schemes under performance, power, energy, area constraints using different processor cache configurations and a collection of representative biomedical workloads;
- Precise analysis of the quantitative data for the evaluated branch-prediction schemes for current and future implant processors; and
- A sound methodology and toolset for selecting best-suited branch-prediction mechanisms for different biomedical (or other) workloads.

The work presented here is original, as compared to related works, in that: i) it studies the whole processor when different prediction schemes are utilized and particularly their reaction to different I/D-cache sizes, ii) it involves 3 more metrics in the study apart from performance, and iii) it targets a different class of low-power devices with particular idiosyncrasies.

5.2.1 Experimental setup

The evaluation environment for this work is almost identical to that set up for studying different cache geometries. Some adjustments have been made to

feature	value
ISA	32-bit ARMv5TE-compatible
Pipeline depth / width	7/8-stage, super-pipelined / 32-bit
RF size	16 registers
Issue policy	in-order
Instruction window	single-instruction
I-Cache L1	*var-size, 2-way assoc. (1-cc hit / 170-cc miss lat.)*
D-Cache L1	*var-size, 2-way assoc. (1-cc hit / 170-cc miss lat.)*
TLB	1-entry
Branch Predictor	**var-type, 4-cc mispred. lat. (32-entry RAS)**
BTB	**var-size**
Write Buffer	2-entry
Fill Buffer	2-entry
Mem. bus width	1B (1 mem. port)
INT/FP ALUs	1/1
Clock frequency	2 MHz
Implem. technology	0.18 μm @ 1.5 Volt

Table 5.6: *XTREM configuration for study on BPRED schemes.*

the aforementioned, Perl-based wrapper to support – apart from varying cache configurations – also different BPRED schemes, as allowed by the XTREM simulator. $10-KB$ ECG data have been used in this case, too, to reflect implant memory sizes properly. All eight ImpBench components have been used as usable workloads. The benchmarks are reported once more in Table 5.5 along with additional statistics pertinent to this study. XTREM has been used for evaluating different BPRED schemes. Its characteristics have been modified accordingly and are summarized in Table 5.6 – processor parameters under investigation are highlighted in bold in the table.

5.2.2 Considered branch-prediction schemes

A large range of branch-prediction techniques has already been proposed in the literature. As previously discussed, our envisioned biomedical-implant processor is being designed - among others - under constraints of ultra-low power consumption and miniature form factor at the calculated cost of limited performance. Accordingly, the processor pipeline will feature a small depth. This set of attributes has effectively narrowed our evaluation effort towards the less complex end of the branch-prediction spectrum.

In the work at hand, we evaluate two static-prediction techniques, i.e. *ALWAYS TAKEN* and *ALWAYS NOT-TAKEN*; it is interesting to investigate how these low-sophistication (but also low-complexity) schemes perform in a

Figure 5.17: *Illustration of a BTB entry in the case of a bimodal predictor (Courtesy: [64]).*

implantable-device context. We further evaluate one dynamic-prediction technique, i.e. an *N-entry, direct-mapped BIMODAL (2-bit)* predictor which is coupled with a Branch-Target-Buffer (BTB) structure used to drive branch penalties down. The BTB stores the history of branches that have executed along with their targets. Figure 5.17 shows an entry in the BTB, where the tag is the instruction address of a previously executed branch and the data contains the target address of the previously executed branch along with two bits of branch-history information (four states: strong-taken, weak-taken, weak-not-taken, strong-not-taken).

There clearly are more sophisticated techniques than a bimodal predictor to achieve higher prediction accuracy in the general case (e.g. skew predictor, gshare predictor) but their complexity is considerably higher, as well. More general, n-bit predictors could be also studied but it has been sufficiently proven that 2-bit predictors score almost as high as infinite-bit predictors [59]. Besides, we would have liked to explore slightly more complex, dynamic predictors, too, but we are hindered by the limited capabilities of XTREM. We did not take the approach assumed by many experts in the field of developing our own standalone BPRED simulation models since we still wish to evaluate the overall effects that different BPRED schemes have on the processor core.

It is important at this point to mention that, for reasons of reliability as well as design complexity, our biomedical processor is meant to feature single-threaded execution, at least in the foreseeable future. Accordingly, all branch-prediction schemes are similarly evaluated on the XTREM simulator on single-executing, non-switched programs. Therefore, the accuracy and performance of the various branch-prediction schemes reported hereafter is pure and not not subject to deterioration due to context switching, as Pasricha and Veidenbaum have shown to occur [108].

BPRED configuration	scheme	BTB #entries
bc01	TAKEN	n/a
bc02	NOTTAKEN	n/a
bc03	BIMOD	2
bc04	BIMOD	4
bc05	BIMOD	8
bc06	BIMOD	16
bc07	BIMOD	32
bc08	BIMOD	64
bc09	BIMOD	128
bc10 (perfect)	BIMOD	4K
L1-cache configuration	**I-cache, 2-way**	**D-cache, 2-way**
cc01 (min)	none	none
cc02	128B	64B
cc03	1KB	512B
cc04	8KB	4KB
cc05	32KB	16KB
cc06 (opt)	64KB	32KB

Table 5.7: *Branch-prediction and I/D-cache configurations used.*

5.2.3 Evaluation study

In this study we have evaluated 10 different branch-predictor configurations, as shown in Table 5.7. The first two are the **always-taken** and **always not-taken**, static predictors. The remaining eight focus on the **bimodal** predictor, as discussed in the previous section, with an increasing number of entries for the BTB. The last configuration utilizes an unrealistically large BTB of 4K-entries - in effect, an infinite BTB - used as a reference for (almost) perfect predictions.

To make the study more involved and identify subtler interactions among the various processor components, we have also chosen, along with the different BPRED configurations, to co-vary also the L1 I/D-cache structures. Based on the previous findings, we have selected two extreme cache configurations (one with no L1 caches and one with optimally-sized L1 caches) as well as four intermediate configuration nodes, as shown also in Table 5.7. The combination of the branch-predictor and cache configurations brings the total number of processor-simulator configurations to 60; that is, for each cache configuration, all predictor configurations have been evaluated. The 8 benchmark applications presented above, have been run for each possible predictor/cache combination and their results have been averaged over all possible configurations.

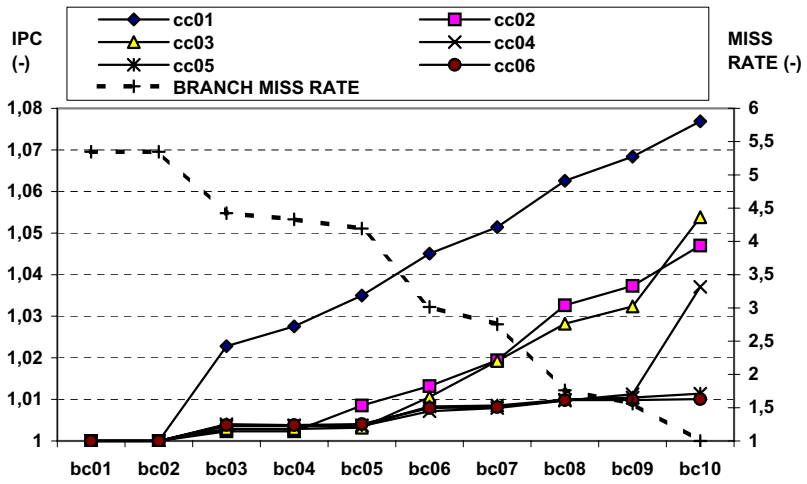


Figure 5.18: *Normalized, average IPCs (left y-axis values) and normalized overall branch miss rate (right y-axis values) for various BPRED/cache configurations.*

In the following discussion, we plot various metrics as normalized values to the *per-configuration* minimum value so as to stress the differences between the various presented schemes. Further, reported plot values in fact are values averaged across all 8 biomedical benchmarks.

With this clarification, we first evaluate the performance of the processor with an improving branch-predictor scheme. In Figure 5.18, average IPCs for all six cache configurations are plotted. Overall, we can see that for our simulated simple and slow ($2 - MHz$ clock frequency) processor, IPC gains with improving predictor schemes are diminishing (up to about 8% w.r.t. the baseline) with increasing cache sizes, although the total branch miss rate drops considerably.

Relatively speaking, cache configurations cc01, cc02 and cc03 benefit the most from improved branch prediction. That is, a processor with larger caches hides the branch misprediction penalties better than one with smaller caches by capturing more instruction fetch requests from main memory. Configuration cc01 (no cache), in particular, displays the most impressive IPC gains compared to the other cache configurations. Inversely, this means that processors with smaller caches ought to benefit the most from an efficient branch prediction (*BPRED*) scheme. Last, we can observe that both static schemes (predictor configurations bc01 and bc02) impact IPC minimally (they form the baseline) and in a similar fashion for all cache sizes while significant speedup is observed

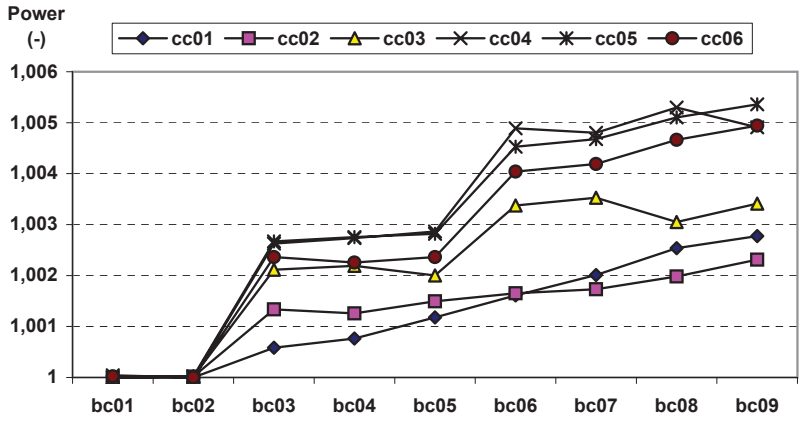


Figure 5.19: *Normalized, average power consumption for various BPRED/-cache configurations.*

from configurations bc05 or bc06 and up for most cache configurations.

The next metric we investigate is the average power consumption. In Figure 5.19, power figures are plotted for all configuration combinations. The bc10 configuration results in excessive power consumption in the BTB component of the processor ranging from 338 mW^2 (or 376% normalized w.r.t. the baseline) for cc01 to 563 mW (or 962% normalized w.r.t. the baseline) for cc06. This is due to its excessive size and has been omitted from the current and following plots to maintain a good resolution for the other nine BPRED configurations.

The main observation in this figure is that smaller-cache processor configurations increase their power consumption at a faster pace with improving BPRED schemes. This observation complies with the IPC plots of Figure 5.18. That is, the higher IPC gains noticed in the presence of smaller caches are sustained by necessary increases in the processor core power consumption. Particularly the IPC bump observed in Figure 5.18 lying in the locus of bc05 to bc06 is followed by a related increase in power consumption across the majority of cache configurations.

The question arises, then, whether the noticed, net performance increase is worth the extra power costs. Figure 5.20 has been plotted to address this ques-

[2]The XTREM simulator has not been initially designed for modeling a μ W-level processor. Thus, the relative differences between the power figures reported here should be considered, rather than their absolute values.

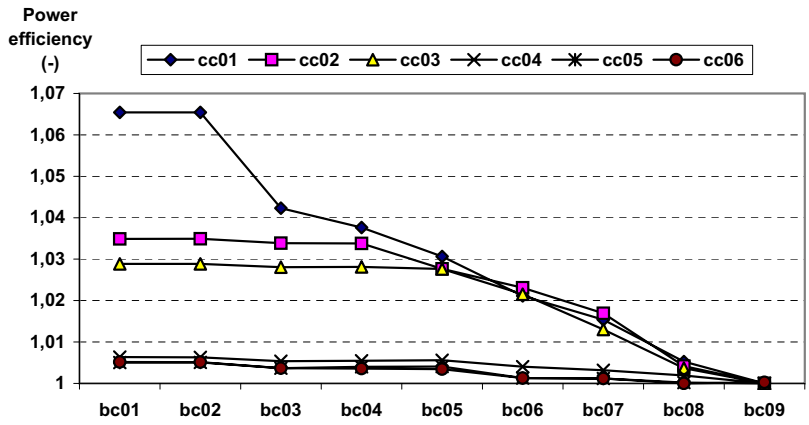


Figure 5.20: *Normalized, average power consumption per instruction per cycle (i.e. instruction efficiency w.r.t. power) for various BPRED/-cache configurations.*

tion; it shows the normalized plots for the power consumed (in mW) per instruction per cycle, i.e. the normalized instruction power efficiency. In this type of figure, "lower values are better"; it is clearly visible that for larger-cache configurations, the impact of more sophisticated BPRED schemes is lower, thus the power savings are similarly smaller. We observe, however, that the power efficiency of smaller-cache configurations is poor but improves more rapidly compared to larger-cache ones.

Last, the minimal impact the two static predictors (bc01 and bc02) have on the IPC is revealed in Figure 5.19, as well. Since they promote instruction-level parallelism (ILP) the least, they also don't stress the core much compared to the other BPRED schemes, resulting in the lowest power profiles across all evaluated schemes.

Apart from average power consumption, for embedded systems with a very constrained energy budget such as implants are, it is also important to examine the overall energy spent by the processor for executing all assigned tasks. Energy, by definition, depends heavily on program execution time and, thus, energy plots are not necessarily identical to average-power plots.

In Figure 5.21 overall energy budgets for different BPRED/cache configurations are plotted. Opposite to the case made on power before, the metric at hand (i.e. energy) achieves a minimum value when moving to more complex BPRED configurations, showing a dramatic improvement from configuration

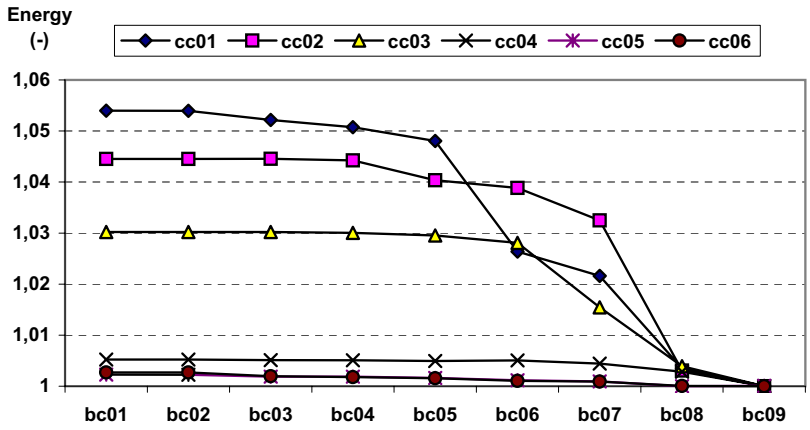


Figure 5.21: *Normalized, total energy expenditure for various BPRED/cache configurations.*

nodes bc07 and bc08 and onwards. When seen from the cache-size perspective, the cache configurations that lead to the highest energy savings are the larger-cache ones (cc04, cc05 and cc06), opposite to the power results. Choosing between a power-efficient and an energy-efficient configuration depends on the priorities imposed on the design specifications of the processor.

In the locus of bc05 or bc06 where IPC has been shown to manifest a non-trivial speedup, energy expenditure does not visibly drop across almost all cache configurations. This hints toward the fact that, at that point, the power costs needed to sustain the higher IPC annul the achieved speedup.

Figure 5.22 plots the instruction energy efficiency of the various configurations. Contrary to the previous figure, this figure presents the same trends as Figure 5.20. The most noticeable difference is that, with the exception of cc01, the lines do not drop in this case as radically around the locus of bc5 and bc06 as in the case of power efficiency.

In this part of our analysis, it also makes sense to consider the area cost of the various BPRED schemes when moving to more advanced techniques. We have, therefore, properly configured and run CACTI v3.0 to collect area-utilization figures for various predictor circuits. Area for the static predictors as well as for the first two bimodal configurations have been based on estimations. CACTI v3.0 (instead of any newer versions) has been used since it is suitable for modeling simpler (older) cache-like structures (such as the BTB) and at an implementation technology identical to the one of the simula-

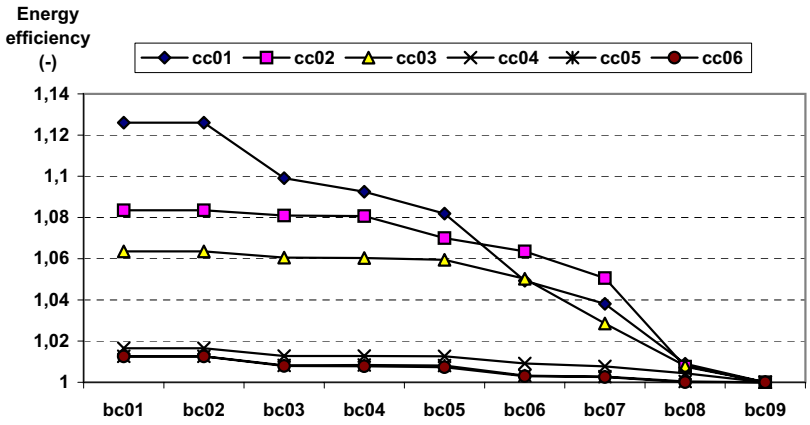


Figure 5.22: *Normalized, total energy expenditure per instruction per cycle (i.e. instruction efficiency w.r.t. energy) for various BPRED/-cache configurations.*

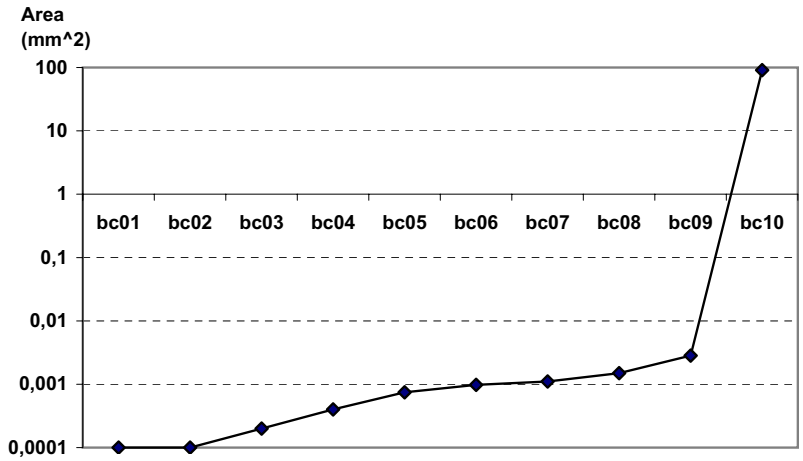


Figure 5.23: *Total BPRED-scheme area (in $\mathtt{mm}^2$) for all BPRED configurations.*

tor ($\mathtt{0.180}\mu\mathtt{m}$). CACTI results are illustrated in Figure 5.23. Please notice the logarithmic scale of the plot.

For selecting the best BPRED configuration for different I/D-cache geometries, we have based our evaluation on performance, power consumption, energy expenditure and area. As a performance metric, we have chosen the IPC

	PRECISION LEVELS					
n	**cc01**	**cc02**	**cc03**	**cc04**	**cc05**	**cc06**
IPC	0.985	0.990	0.985	0.995	1.000	1.000
power	0.990	0.990	0.990	1.000	1.000	1.000
energy	0.990	0.990	0.990	1.000	1.000	1.000
area	0.980	0.985	0.980	0.995	0.995	0.995

Table 5.8: *Precision levels [0.000: worst, 1.000: best] for IPC, power, energy and area in objective function* (5.1) *for all cache configurations when area is considered.*

instead of the branch miss rate since we do not wish to study the BPRED techniques in an isolated environment but, rather, we wish to capture overall system performance as a function of predictor type. That is why we have used a processor (rather than branch-predictor) simulator as our testbed. For the very same reason we have also used total average power consumption and total energy budget as our second and third metric, respectively.

To find optimal BPRED schemes for each cache configuration, we have assumed approach identical to the one taken for evaluating different cache geometries: We have employed formula 5.3 as our **objective function** for *minimization*. Table 5.8 shows, per cache node, the precision levels we have used for all metrics. Remember, that the closer to '1.000' a metric's precision level is, the more strictly constrained the objective function (5.1) becomes and the more close to optimal is the solution. In the table above, precision levels have been iteratively decreased until a single solution (i.e. the optimal) to each objective function was found.

In the absence of accurate processor design constraints[3] we have imposed an intuitive ordering to the metrics used in the objective function. In order of decreasing importance, the metrics have been ranked: power and energy first, then IPC and, last, area. It is with this order that we have adjusted the precision levels in search of the best solution for all six objective functions.

We can readily see from Table 5.8 that, as we move from configuration cc01 to cc06 (i.e. to higher cache sizes), the precision levels become increasingly higher, that is, stricter. This indicates that with increasing cache sizes, optimal values across the four metrics are less scattered; thus, the solution is more straightforward.

[3] Previous, related work does not provide solid data for properly adjusting the metric weights, i.e. the contribution of the different metrics, in formula (5.1). To avoid unfair biasing of the results, we have assumed here a policy of equal weights.

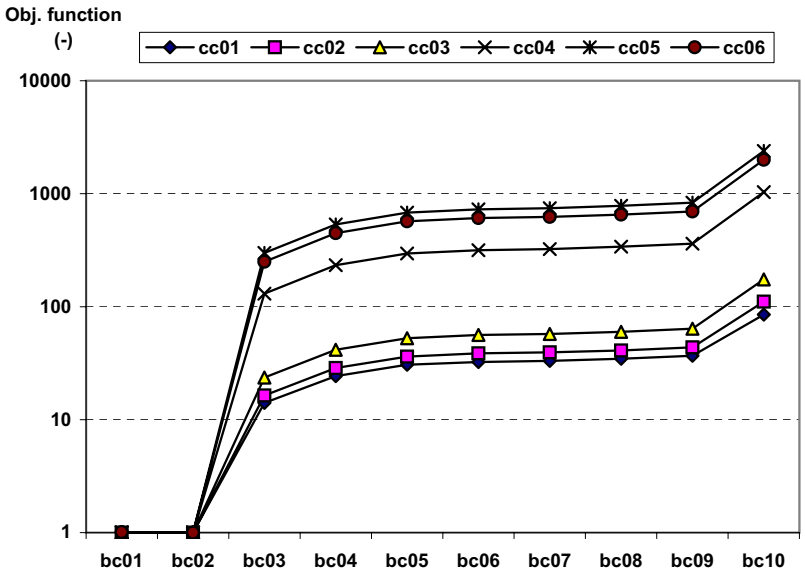


Figure 5.24: *Normalized results to the minimum value for various cache configurations of objective function* (5.1) *when area is included.*

The results of the optimization process are graphically depicted in Fig.5.24. With deviations of 1% or less between them, the static predictors ALWAYS TAKEN and ALWAYS NOT-TAKEN minimize the objective function across all cache configurations. The remaining configurations follow with significantly worse ranking, although we can see that smaller-cache configurations would benefit more from the bimodal predictors of any size, compared with larger-cache configurations.

Some commenting on this result is needed here. In the objective functions above we have included the area metric. However, area calculation has been done through CACTI, outside the XTREM simulator. As a result, the area utilization for specific BPRED configurations as well as the modeling of the BTB through CACTI has been based, to a certain degree, on speculations and assumptions.

For fear of skewing the optimization results and for purposes of completeness we have also optimized the objective function (5.1) once more after disregarding the area metric $A_{PD}(x)$. In so doing, the precision levels of the remaining metrics have also been readjusted, as illustrated in Table 5.9. The new precision levels are somewhat lower than before to compensate for the area vari-

	PRECISION LEVELS					
n	**cc01**	**cc02**	**cc03**	**cc04**	**cc05**	**cc06**
IPC	0.980	0.980	0.990	0.995	0.999	0.999
power	0.980	0.990	0.990	0.999	0.999	0.999
energy	0.980	0.990	0.990	0.999	0.999	0.999
area	0.000	0.000	0.000	0.000	0.000	0.000

Table 5.9: *Precision levels [0.000: worst, 1.000: best] for IPC, power, energy and area in objective function* (5.1) *for all cache configurations when area is not considered.*

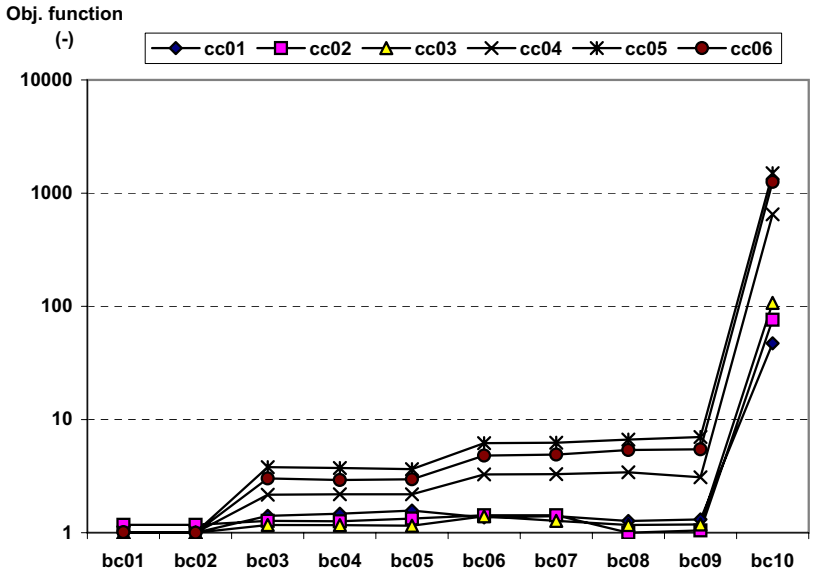


Figure 5.25: *Normalized results to the minimum value for various cache configurations of objective function* (5.1) *when area is not included.*

able, yet they exhibit the same trend as in the previous case; that is, they can be raised higher with increasing cache sizes.

Graphical depictions of the objective-function results in this case are shown in Fig.5.25. Most suitable BPRED configurations for almost all cache sizes are, as before, the ALWAYS TAKEN and ALWAYS NOT-TAKEN static schemes. However, objective-function trends are general different, in this case. We can observe that for smaller-cache configurations the objective function is not monotonously increasing but, rather, features more than one (local) minimum. As a result, when the area metric is factored out of the equation, more solutions

than the TAKEN/NOT-TAKEN schemes may become attractive, especially for smaller-cache configurations. To exemplify, for cache configuration cc03 the optimal BPRED configuration is in fact bc08, i.e. a bimodal predictor with a 64-entry BTB, followed by the typical TAKEN/NOT-TAKEN schemes. These suboptimal configuration nodes could become the nodes of choice in a design where e.g. a lower threshold on IPC is imposed; thus, in cases of constrained optimization.

Overall, and to combine the results of both Fig.5.24 and Fig.5.25, it becomes clear that - even by factoring the area component out of the optimization function - the cost of bimodal predictors of any BTB size is too high to justify the overheads incurred to the processor, across all cache sizes. Practically speaking, this means that the reduction in branch mispredictions offered does not improve performance to the point that introduced power penalties of a dynamic predictor can be justified.

5.2.4 Conclusions

We have provided a detailed investigation of various branch-prediction configurations in conjunction with different I/D-cache configurations, tested on a specially modified, low-power, cycle-accurate processor simulator. Findings indicate that, under (relaxed) area constraints, the optimal selections for branch prediction interchangeably are the static schemes ALWAYS TAKEN and ALWAYS NOT-TAKEN, regardless of processor cache size. This means that for slow-performing, ultra-low power processors and the given biomedical workloads, dynamic-prediction schemes are too expensive to implement, their drawbacks outperforming their benefits.

A second interesting result, however, is that processor configurations with smaller caches (i.e. for I/D-cache sizes up to $1024KB/512KB$, respectively) benefit from efficient BPRED schemes more. Especially under relaxed area constraints, more suboptimal configurations but close to the optimal one exist. In lack of more accurate data from the literature, in this work we have imposed loose design constraints on the evaluated metrics. Yet this last observation can be particularly useful under highly constrained processor design.

5.3 Summary

In this chapter, with implant benchmarks available, input data to drive them and a suitable simulator testbed, we were able to perform two in-depth inves-

tigations on two particular microarchitectural aspects of the envisioned SiMS processor, namely, the cache and branch-prediction subsystems. Our choice for studying these two subsystems was dictated primarily by their significance in the characteristics of the processor as well as by the limited capabilities provided by XTREM. Except for the novel application field, what makes this investigation more important is the fact that we studied the effects of various configurations of the two subsystems (and their interplay) with respect to the whole processor core. Total performance, power, energy and area metrics have been utilized to get the complete picture when considering such subsystems in an implant processor (or any processor, for that matter). Truly enough, the investigation findings have generated interesting and – in some cases – counter-intuitive conclusions.

Last but not least, the work presented in this chapter had another side-benefit: The vast needs in automated simulation runs needed for this study have driven us to extend the XTREM simulator with a Perl-based wrapper construct. This wrapper, through interfacing to CACTI, an established simulator for cache-area estimations and various Bash (shell) scripts, has enabled automatic configuring of the simulators, feeding them with any number of benchmark-dataset combinations, running a large number of simulations unattended, collecting, aggregating and reporting data as the ones we showed in the previous sections. This methodology can as well be used, with minor or moderate modifications, for other (micro)architectural experiments.

Note. The content of this chapter is based on the following papers:

C. Strydis, **Suitable Cache Organizations for a Novel Biomedical Implant Processor**, 26th IEEE International Conference on Computer Design (ICCD'08), pp. 591-598, Lake Tahoe, California, USA, October 2008.

C. Strydis, G. N. Gaydadjiev, **Evaluating Various Branch-Prediction Schemes for Biomedical-Implant Processors**, 20th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP'08), pp. 169-176, Boston, MA, USA, July 2009.

6

Automated exploration of SiMS-processor microarchitectures

OPTIMAL cache and branch-prediction subsystems for the SiMS processor have been studied in the previous chapter. These studies have offered many design insights on the processor, yet provide local and, by necessity, *biased* optimizations.

On the other hand, it is well-known that the global optimization of multiple design objectives – such as performance, power and area – across *all* processor subsystems is a non-trivial task. One must explore all possible processor configurations, compute the corresponding design objectives and find the *Pareto-dominant* solutions to be included in the final trade-off set. Since, typically, the design parameters that affect a processor are numerous, computing the behavior of all their possible combinations is quite hard, if not impossible. For example, by considering 13 processor *design parameters* represented by 36 (binary) bits, if one were to simulate all combinations, one would need to evaluate 2^{36} = 68,719,476,736 different processor configurations to identify the true *Pareto front* – an unrealistically high number. To make matters worse, in this new field of implant processors there is no established set of processor characteristics that would allow meaningful limiting of the above number of potential configurations.

We, therefore, need a way to realistically approximate this ideal trade-off set without performing all possible simulations. Furthermore, even in an approximated scenario, thousands of configurations might still have to be evaluated and compared to get a good approximation of the true Pareto front. Hence, we need an automated method for doing so.

In this chapter, we build upon the methodology developed in the previous chapter and introduce ImpEDE, a new tool offering *automated*, *multiobjective* DSE

of optimal SiMS processor *microarchitectural* configurations. Through ImpEDE, we introduce one more optimization goal – *execution time* – next to performance, power consumption, energy expenditure and area. The need to introduce the notion of "hard deadlines" in program execution have coerced us in developing an updated version of ImpBench (v1.1), reported next. As a last and culminating point in this thesis, we utilize ImpEDE, ImpBench v1.1 and suitable implant applications extracted from the survey in Chapter 2 to offer a number of optimal SiMS-processor solutions.

6.1 ImpEDE: A DSE tool for implant processors

To the best of our knowledge, none of the existing DSE tools are explicitly concerned with implantable systems. The field of biomedical, microelectronic implants is new and fast-progressing and calls for particular design constraints such as *ultra-low power consumption*, *high fault-tolerance levels* and *tight execution deadlines*. What is needed is a fresh top-down approach to the field where implant applications are extensively profiled in a properly fine-tuned environment and the findings are used to drive an (automated, if possible) design-exploration effort for a suitable implant device. Setting up such an environment is a non-trivial problem as its specific parameters are either unknown or undisclosed, subject to tight proprietary controls.

In this work we present a novel framework intended for the systematic, rapid and adaptable DSE of processor architectures suitable for biomedical implants. The framework is termed **ImpEDE** (**Imp**lant-processor, **E**volutionary, **D**esign-space **E**xplorer) and provides careful investigation of the processor design space through the use of a particular *genetic-algorithm* (GA) variant called NSGA-II, along with cycle-accurate simulations, considering realistic design constraints imposed by our prior knowledge of the field. This implementation has exhibited – due to the large problem complexity – very long computational times and, therefore, a parallelized version of the algorithm has also been implemented and described here. Also, we attempt to fine-tune ImpEDE to the goal at hand by providing the first – in our knowledge – tool to offer bounded DSE. Concisely, the contributions of ImpEDE are:

- A first yet educated attempt towards the systematic, automated and accurate design of implant processors;
- A fine-tuned toolset that delivers optimized implant-processor configurations across multiple first-order (e.g. performance, power) and second-

order (e.g. hard real-time deadlines) objectives; and

- A freely available parallelized version that can be expanded with additional design objectives and constraints and extended to applications classes other than implants.

6.1.1 Framework organization

Before introducing the DSE framework, we first have to identify the nature of the problem we attempt to solve: In designing our implant processor, we have formulated our problem as a multiobjective-minimization problem, with the objectives being processor *latency*, *average power consumption* and *area cost*. This is a set of first-order objectives typically optimized for in digital design. In the future and as the framework matures, we wish to add more objectives in our design effort such as *fault coverage* and more constraints such as *hard realtime deadlines*. In later sections, we will exemplify the latter by imposing a hard deadline on execution time (i.e. latency).

Since our objectives are minimizing area and power while maximizing performance, in order to convert our problem to a fully minimizing problem, we need to take the complement of performance as the objective encoded in our framework. We use IPC as the metric of performance. Therefore, we can simply use IPC*(-1) as the objective to be minimized[1]. For the rest, we use the metrics mm^2 and mW as the area and the power objectives, respectively.

With these objectives in mind, we have designed the multiobjective, DSE framework shown in Figure 6.1. At first, the selected GA (NSGA-II) generates valid processor configurations (i.e. a set of parameters) – also known as "chromosomes" – that are fed to the XTREM cycle-accurate, performance and power simulator and to the CACTI cache-area simulator. XTREM also accepts as inputs implant-related benchmarks and assorted datasets (ImpBench). Then, both simulators execute and their resulting performance, power and area figures are fed back into the waiting GA which uses them to evaluate the *optimality* of the currently simulated processor configuration. This process is repeated a number of times equal to the preset chromosome *population*; then, a few best-performing chromosomes – based on their *fitness* results – are selected, processed and propagated to the next round of optimizations, also known as *generation*. With each successive generation, increasingly better chromosomes are found and promoted; that is, we are approaching the true Pareto front for our

[1]Note that cycles per instruction (CPI) could have also been used: since CPI is the inverse of IPC, it would give an identical relative ordering of processor configurations as IPC*(-1).

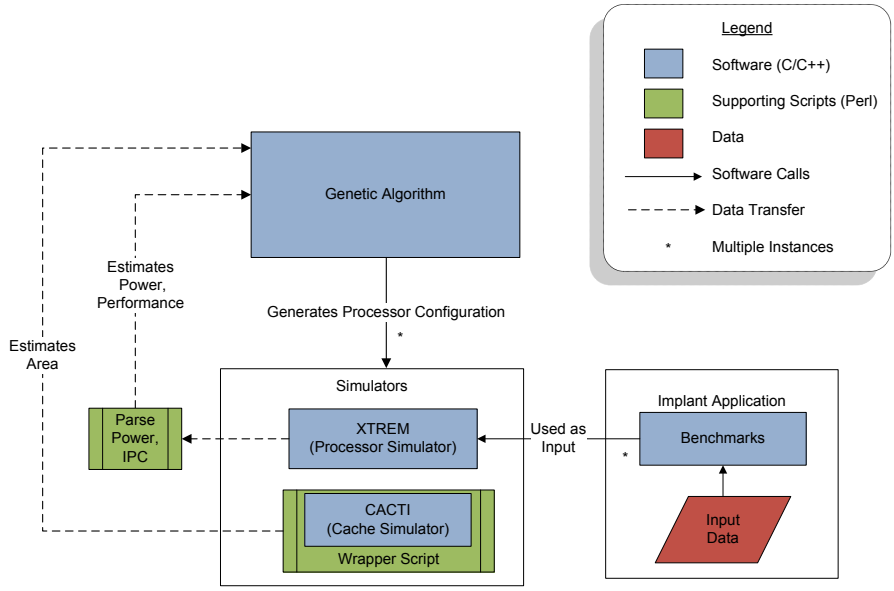


Figure 6.1: *Framework organization.*

DSE problem. Figure 6.2 shows the progress of the front at various evolution stages for a particular run of our framework. In the following subsections, we will describe in more detail the components of the framework as well as the choices made on the GA parameters such as the population size, the number of generations and the chromosome-selection policy used.

6.1.1.1 Genetic algorithm: NSGA-II

The classical single-objective optimization methods can be used to perform multiobjective optimization by reformulating the multiobjective optimization problem into a single-objective one. This can either be done by combining the objectives into a single aggregate objective [75] or by only considering one of the objectives and moving the rest to a constraint set [88]. When designing this framework, we first used a single-objective GA that employed the weighted-sums approach for finding the fitness of an individual. However, this was quickly abandoned as we could not logically assign the values of the weights since there was no rationalization for preferring one objective over another without more information about the problem domain. Besides, there was no way of knowing the absolute upper and lower limits of the three objectives

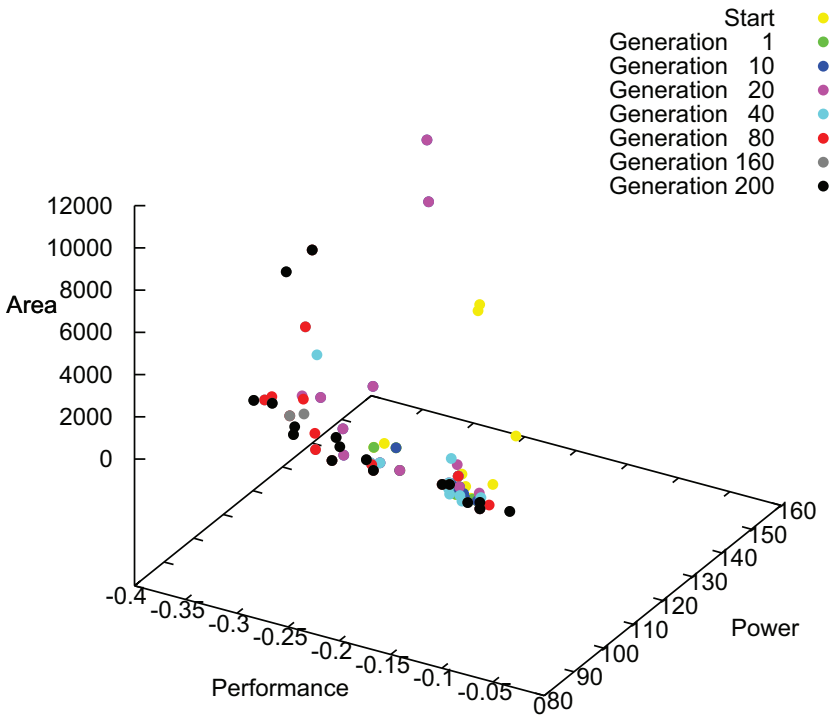


Figure 6.2: *ImpEDE-generated Pareto solutions (i.e. implant-processor instances).*

(performance, power, area). Finally, such approaches suffer from being unable to find solutions to problems having non-convex fronts. Therefore, special algorithms were formulated for multi-objective problems [30].

NSGA [129] was one of the first GAs that evolve Pareto-front solutions to multi-objective problems. NSGA-II [31] is the successor of NSGA that overcomes some of the limitations faced by the former. NSGA-II evolves Pareto fronts using an elitist approach and uses density and crowding distance metrics to ensure well spread out points along the front, at the same time having a lower computational complexity than its predecessor. It has, therefore, become in its own right widely accepted and used in diverse applications such as [114] and [14]. Due to its superiority over other algorithms, popularity, ease of use and availability, we use it as our algorithm of choice.

feature	value
ISA	32-bit ARMv5TE-compatible
Pipeline depth / width	7/8-stage pipeline / 32-bit
RF size	16 registers
Issue policy	in-order
Instruction window	single-instruction
I-Cache L1	VAR size&assoc. (1-cc hit / 170-cc miss lat.)
D-Cache L1	VAR size&assoc. (1-cc hit / 170-cc miss lat.)
BTB	VAR size, fully-assoc. / direct-mapped
Branch Predictor	VAR (4-cc mispred. lat.)
Ret. Address Stack	VAR size
I-TLB	1-entry / 1-entry
D-TLB	1-entry / 1-entry
Write Buf. / Fill Buf.	2-entry / 2-entry
Mem. bus width	8-bit (1 mem. port)
INT/FP ALUs	1/1
Clock frequency	2 MHz
Implem. technology	0.18 μm @ 1.5 Volt

Table 6.1: *XTREM configuration for ImpEDE integration.*

6.1.1.2 Processor & cache simulators

In the current version of our DSE framework, evaluation of the performance and power consumption of a given chromosome (i.e. processor configuration) has been based on the XTREM simulator. We have extensively used XTREM in our previous studies on the implant processor and, in order to match our application field better, we have disabled many of XTREM's architectural parameters. In this case, however, we wish to allow for some degree of freedom in the processor design parameters so that the GA can explore a wider range of possible configurations. We have, thus, ended up with the XTREM configuration summarized in Table 6.1[2]. In the next section, we shall go through the selected simulator parameters and the way they have been encoded in the GA. It should be noted that flexible *wrapper* scripts (based on the methods developed previously, see Chapter 5) have been used to provide the input to and capture the output of XTREM. As a result, the internal framework structure has been kept highly *modular* allowing for porting faster, more accurate or more powerful simulators in the future.

For evaluating the area cost of each chromosome, we have made the valid approximation that the subsystem dominating our envisioned implant processor

[2]Values denoted with 'VAR' indicate adjustable parameters by the GA.

Benchmark type	Name	Size (KB)	#Instr.*	Sim. time* (sec)
Compression	miniLZO	16.30	199,163	3.07
	Finnish	10.40	852,663	21.82
Encryption	MISTY1	18.80	1,268,465	16.65
	RC6	11.40	864,930	9.37
Data integrity	checksum	9.40	62,869	0.80
	CRC32	9.30	419,159	5.46
Real applications	motion	9.44	859,371	31.16
	DMU	19.50	36,808,268	37.25

Table 6.2: *ImpBench components. Columns denoted with a (*) indicate average values for* $1 - KB$ *input datasets.*

Dataset name	size (Bytes)	Samples (#)	duration (sec)
Electromyogram II (EMGII)	1147 / 9605	144 / 1201	0.288 / 2.402
Electroencephalogram (EEGI)	984 / 9616	123 / 1202	0.615 / 6.010
Electrocardiogram (ECGI)	912 / 9615	114 / 1202	0.114 / 1.202
Respiratory Cycle I (RCI)	1192 / 9520	149 / 1191	1.490 / 11.910
Pulmonary Function I (PFI)	1184 / 9240	148 / 1155	1.480 / 11.550
Skin Temperature (AEP)	1120 / 9736	140 / 1217	0.700 / 6.085
Blood Pressure (BP)	1128 / 9545	141 / 1198	0.282 / 2.396

Table 6.3: *Physiological input datasets with double-precision (8-Byte) data samples of sizes 1-KB and 10-KB.*

is the cache (which holds also true for modern general-purpose processors). Furthermore, as can be seen from Table 6.1, more adjustable parameters (e.g. BTB, WB, FB) include some cache-like structure in them. Therefore, for quantifying each chromosome's area cost, we have used CACTI, a well-known, cache-area estimation tool. CACTI v3.2 has been primarily used since it is suitable for modeling simpler (older) cache-like structures (such as the BTB) and at an implementation technology identical to the one of the simulator ($180\ nm$). In any case, the wrapper scripts we have created (Figure 6.1) can also handle CACTI versions 4.1 and 6.0, if desired.

6.1.1.3 Biomedical benchmarks & input datasets

In consistency with our prior experiments, the eight ImpBench components have been used as representative implant workloads for execution on XTREM, for evaluating different chromosomes. These benchmarks represent anticipated common tasks running on future implant processors and exhibit varied characteristics, as shown in Table 6.2.

Furthermore, we have considered actual physiological input data that has been provided from the BIOPAC (R) Student Lab PRO v3.7 Software. A concise overview of the dataset details is provided in Table 6.3. Each chromosome represents a particular processor instance onto which each all benchmarks except *DMU* (which runs on a single, hard-coded input) are fed with each of the 7 datasets (of (1 *KB* or 10 *KB*)) and are executed. This accounts for a total of 50 benchmark runs for the 1 *KB* case and another 50 for the 10 *KB* case. As we shall see in Section 6.1.1.4, this is a substantial amount of (cycle-accurate) simulation time. In order to get practical results in our limited time frame and *without loss of generality*, for this work we have selected and executed only the EMGII dataset (both sizes) as it displays *worst-case* performance characteristics. In so doing, we have limited the number of runs to 8 per dataset size. We, nonetheless, explore the design space for both sizes in order to investigate the effect dataset size has on the Pareto-optimal solutions.

6.1.1.4 Parallelization & optimization

As shown in Table 6.2, evaluating a single processor configuration (i.e. one GA individual) with a single input (EMGII) and a single data size (1 *KB*), across all 8 benchmarks takes 125.58 seconds, on average. Assuming the $10 - KB$ datasets run $\times 10$ as slow as the $1 - KB$ benchmarks (except for DMU) and, considering optimization across all 7 dataset types, a full run of the GA with a population size of 20 and 200 generations will take approximately 343 **days** per result. We do not consider the GA run-times in this calculation as the execution overhead of the parallelized GA was found to be negligible, contributing only 0.13 seconds per generation.

Since this run-time (a little less than a full calendar year) is quite prohibitive, we parallelized the evaluation stage of the GA so that different individuals are evaluated on idle CPUs of the CE-group's laboratory machines. Hence, the speedup offered by our parallelized version is equal to $\sim P/\lceil P/N \rceil$, where P is the population size and N is the number of computers available. During the run-time of the GA, support scripts periodically search for and prepare free machines for the algorithm to use; these machines are used on the lowest priority in order to not disrupt regular usage. Therefore, this framework is *expandable*, *modular* and requires *minimal dedicated resources*.

It turns out that, at any given time, we had around 20 machines available for computation. Therefore, the runtime has been effectively reduced to about 17.5 days for each run. As discussed before, we reduced this time further by only considering the worst-case input for each benchmark, EMGII, and we

performed separate runs for each of the input sizes of 1 *KB* and 10 *KB*.

6.1.2 Framework fine-tuning

In the previous section, we went through the various building blocks of the DSE framework. Adjusting the framework to target implant processors requires, however, fine-tuning of the GA parameters and proper encoding (i.e. representation) of the chromosomes. In what follows, we go through such details that make our framework suitable for implant-processor design.

6.1.2.1 Chromosome encoding & XTREM errata

Since GAs optimize the information encoded in the chromosomes, we needed to define a chromosomal representation for the processor parameters that the GA can work with. There are several chromosomal encoding strategies, the simplest being encoding each variable as a string of 1 and 0 bits. In this work we chose this encoding rather than *real encoding* [153] since the processor parameters encoded are integer values. Each chromosome is encoded as shown in Table 6.4. The table lists the processor variables chosen, their ranges as well as the encoding and decoding rules. The included variables are in agreement with the ones in Table 6.1.

The processor *parameters* we chose to include in the search space depended both on what we wanted out of the processor and the capabilities and limitations of the simulators we had. As can be seen from Table 6.4, clock frequency has been encoded but was not used in this version of the GA; we found that running the simulator with different clock-frequency values did not affect the results. For the purposes of this work, XTREM runs on a default clock frequency of 2 MHz, typical for implant processors.

The Write Buffer and the Fill Buffer included in XScale (and, thus, XTREM) help achieve better performance by hiding memory-access stalls when the core is running at a high clock frequency (e.g. 200 *MHz*). This is hardly the case for an implant processor; therefore, we did not encode the two buffers but rather fixed their sizes at the minimum supported by XTREM, i.e. exactly two entries. For similar reasons, Translation Lookaside Buffers (I/D-TLBs) have been excluded from the GA and fixed to a single-entry structure each.

As can be seen from Table 6.4, I-cache and D-cache structures have also been encoded in the chromosome. Although the simulator theoretically supports adjustable I-cache and D-cache latencies, we found that varying the I-cache la-

tency above the default value of 1 often had the simulator hanging. Therefore, we only include D-cache latency as a variable, which we vary from 1 to 16 clock cycles. Note also that the above discussion pertains only to L1 caches. At a stage where embedded applications hardly have even an L1 cache, we think having L2 caches would be an overkill, and therefore exclude them from our exploration. This is also in agreement with the results in Chapter 5.

Except for the processor parameters, also their *ranges* have been defined: i) by the minimum and maximum allowed range that the simulators could support – so that we could capture as much of the design spectrum as possible –, and ii) by the (micro)architectural profiling studies we have performed so far.

6.1.2.2 Population size

The size of the population represents the maximum number of Pareto points the algorithm can find. If we pick too small a size for the population, we would have lesser tradeoffs available. On the other hand, the GA is $O(mN^2)$, where m is the number of objectives and N is the population size. Keeping these factors in mind, we chose a population size of 20, which also coincided with the number of machines we expected to be free at any given time. Therefore, the entire population could be evaluated at once in parallel.

6.1.2.3 Number of Generations

The number of generations represent the time the GA is allowed to reach the Pareto front. We observed that the GA converges rather rapidly to a front, then tries to increase its spread in the subsequent generations. After this, the GA reaches a sort of 'stable state' where it is unable to improve the spread without degrading the distance from the front, and vice versa. We can limit the number of generations at this phase, in order to reduce computation time. We use a reduced problem set - that of only optimizing the *checksum* benchmark, and run this for a large number of generations (1000) in order to approximate the number of generations needed, then use this as a guide to selecting the number of generations for the full problem.

Parameter Name (Ref)	Range	Encoded Range	Bits	Decoding Formula	Remarks
Core Clock Frequency (*Freq*)	[1...64]	[0...63]	6	$n + 1$	Not used in current version
Branch Prediction (*Bpred*)	*Bimodal*, *Taken*, *notTaken*	[0...2]	2	-	bit_0 = *isBimod*, bit_1 = *isTaken*
Branch Target Buffer: Number of Sets (*btb_nsets*)	[32...128]	[0...5]	3	2^{n+5}	Only valid for *isBimod* = *TRUE*
Branch Target Buffer: Associativity (*btb_assoc*)	[1...32]	[0...5]	3	2^n	Only valid for *isBimod* = *TRUE*
Branch Prediction: Return Address Stack (*RAS*)	[0...8]	[0...4]	3	$floor(2^{n-1})$	-
L1 I/D-Cache: Number of Sets (*I/D_nsets*)	[1...8192]	[0...13]	4	2^n	-
L1 I/D-Cache: Block Size (*I/D_bsize*)	[8...32]	[0...2]	2	2^{n+3}	-
L1 I/D-Cache: Associativity (*I/D_assoc*)	[1...32]	[0...2]	3	2^n	-
L1 I/D-Cache: Replacement Policy (*I/D_repl*)	*f*, *r*, *l*	[0...2]	2	-	bit_0 = *isFifo*, bit_1 = *isRandom*
L1 D-Cache: Latency (*D_latency*)	[1...16]	[0...4]	3	$floor(2^n)$	-
L1 I-Cache: Latency (*I_latency*)	[1...16]	[0...4]	3	$floor(2^n)$	Not used in current version

Table 6.4: *Processor design parameters considered in this work, encoded as 36 chromosomal bits.*

For quantifying the *distance* of the solution front Q from the 'true' Pareto front P^*[3] we have chosen Veldhuizen's **Generational-Distance** (GD) metric [142]:

$$\text{Let} \qquad d_i = \min_{k=1}^{|P^*|} \sqrt[p]{\sum_{m=1}^{\mathbb{M}} (f_m{}^{(i)} - f^*{}_m{}^{(k)})^p}$$

$$\text{Then,} \qquad GD = \frac{\left(\sum_{i=1}^{|Q|} d_i{}^p\right)^{1/p}}{|Q|}, \tag{6.1}$$

where Q is the solution under consideration; P^* is the reference Pareto-front, $\mathbb{M}$ is the total number of objective functions, and $f_m{}^{(x)}$ and $f^*{}_m{}^{(x)}$ are the m$^{\text{th}}$-objective-function values of the x$^{\text{th}}$ solutions of Q and P^*, respectively. The lower the value of GD, the closer the distance between Q and P^* and the better the solution, with $GD = 0$ for solutions lying on the reference front.

For quantifying the *diversity (spread)* of the front Q, we have used **Deb et. al's spread metric** Δ [30]:

$$\Delta = \frac{\sum_{m=1}^{\mathbb{M}} d_m^e + \sum_{i=1}^{|Q|} |d_i - \bar{d}|}{\sum_{m=1}^{\mathbb{M}} d_m^e + |Q|\bar{d}} \tag{6.2}$$

where d_i is the same distance metric described in GD, and $\bar{d}$ is their mean and d_m^e is the distance between the extreme solutions of P^* and Q with respect to the m$^{\text{th}}$ objective.

We found both metrics to be very noisy, especially the diversity metric. Therefore, to make them easier to compare, we smooth the data using a moving average with a span of 20 generations. Figure 6.3 shows the resulting metrics. We observer that GD declines rapidly until about the 100th generation, after which it fluctuates around $GD = 0.4$ for a long time until finally dropping to zero around generation #600. The rapid decline is of course due to the solution fronts converging towards the reference. After generation #51, it can be seen that the general trend is that as spread decreases, distance increases and vice versa. The behavior from generations #100-#593 are also expected - as the algorithm searches for new solutions, the distance fluctuates. A minor rise in

[3]Since we do not know the true front (by the problem definition itself), we approximate it by computing a combined front consisting of mutually non-dominating points from the results of 10 separate runs of the algorithm. We call this the 'reference front'.

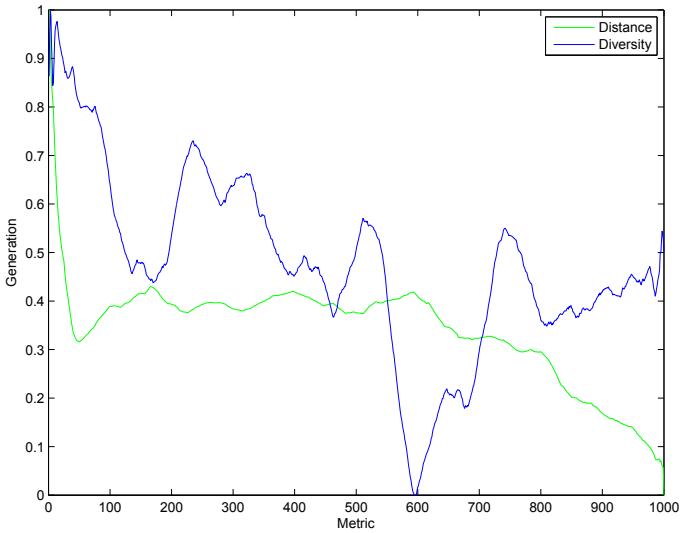


Figure 6.3: *Smoothed distance and diversity metrics over 1000 generations (Benchmark: checksum)*

both GD and spread may not mean that the front is actually further away than the one previously – since we have finite points in the reference front, points that are the same distance from the *actual* front may give slightly varying distance values from the reference front. The behavior at the tail end reflects the fact that the reference solution is a combination of several solutions, and therefore also contains the final solution of the run in question. The apparent rapid convergence seen therefore, is in fact the algorithm moving towards its own final solution – a foregone conclusion. Therefore, we do not consider this region in our analysis.

Since after generation #100, the algorithm seems to oscillate between improving spread and distance at the cost of the other; without loss of precision in both quantities at the same time, we can stop the algorithm around generation #100. Since GAs are random by nature, in order to be sure of getting good results, we run every subsequent experiment for twice this time, i.e. 200 generations. This also compensates for the smoothing we performed.

6.1.2.4 Mutation

There are a number of ways to select the mutation probability, including complicated adaptive mutation strategies [67]. However, the simplest and most

recommended strategy is simply setting the mutation probability as $p_m = 1/n$ where n is the chromosome length [9]. This means that a single bit per chromosome is expected to change from the parent to the child population. Therefore, the child chromosome is likely to vary from the parent chromosome in exactly one attribute, that too by a hamming distance of one. We use this approach for setting the mutation probability in our implementation of the GA, i.e. $p_m = 1/36$.

6.1.2.5 Crossover probability

Crossover allows chromosomes to exchange information that may lead to better chromosomes that combine the "good qualities" of the parent chromosomes. Used carefully, crossover can lead to much quicker evolution times, and is central to the idea of Genetic Algorithms examining more solutions in the more promising regions of the solution space [63]. Therefore, it is important to set a good *crossover probability* P_c, which determines the percentage of chromosomes that undergo recombination at each generation. As in the case of number of generations, we used a reduced problem set – running only *checksum* with the $1 - KB$ EMGII input for 200 generations with different crossover probabilities. Figure 6.4 shows the two metrics for the Pareto fronts resulting from each value of P_c [4]. Keeping in mind the discussion from Section 6.1.2.3, we see from the graphs that $P_c = 0.2$ and $P_c = 0.6$ seem to lead to the fastest convergence and best values for the two metrics over the course of the generations, and therefore $P_c = 0.2$ is chosen for subsequent runs.

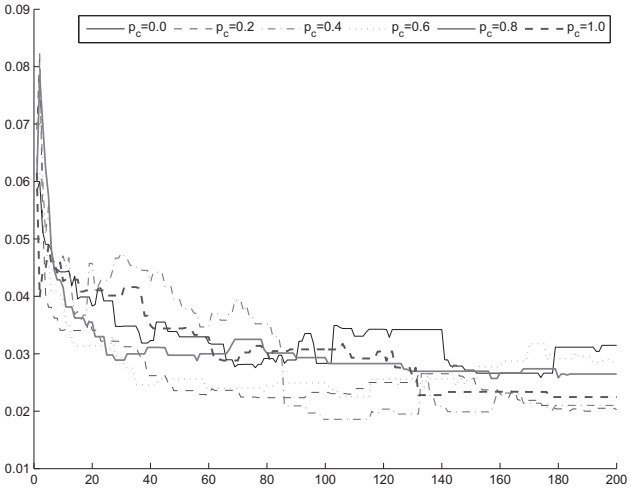
6.1.3 Selected results & validation

In this section, the correct functionality of the framework is demonstrated. Also, an expansion of the framework with a *hard realtime deadline* leading to *constrained design* is illustrated.

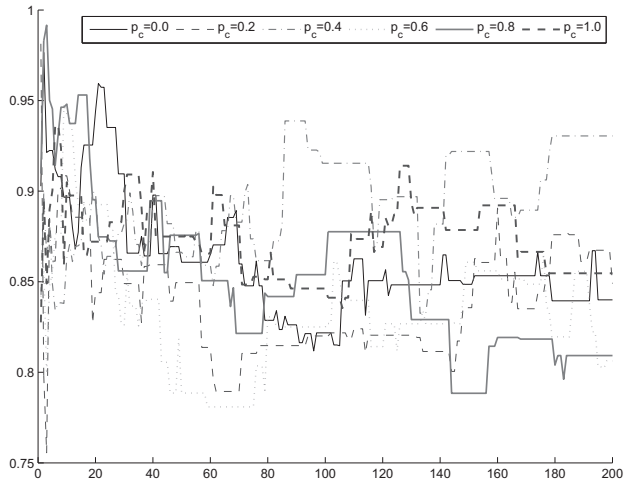
6.1.3.1 Implant-processor results

Having prepared and fine-tuned the framework to the best of our knowledge, we move on to testing it by running it with all the benchmarks, once for each of

[4] Note that in this case, these are the un-normalized, un-smoothed metrics. They appear less noisy due to the fact that the number of generations plotted is much lower than in Section 6.1.2.3.



(a) *Distance metric*



(b) *Diversity metric*

Figure 6.4: *Distance and Diversity metrics for various crossover probabilities (P_c) over 200 generations (Benchmark: checksum)*

the two dataset sizes. We call these the *baseline* results. Figure 6.5 shows the projections of the Pareto front evolved on the 3 Cartesian planes (performance, power, area).

We readily see in both rows of results that the algorithm reaches the "front"

by generation #40. After this, the points spread out and a wider Pareto front is found. We see from Figure 6.5b and Figure 6.5d that the $10-KB$ datasets have a wider front w.r.t. performance. We anticipate this to be so because the bigger dataset increases processor utilization by allowing the caches and branch predictor table to fill and, hence, minimizing processor stalls. This higher performance also leads to a corresponding increase in the power consumption as can be seen from the power axes in Figure 6.5b and Figure 6.5c.

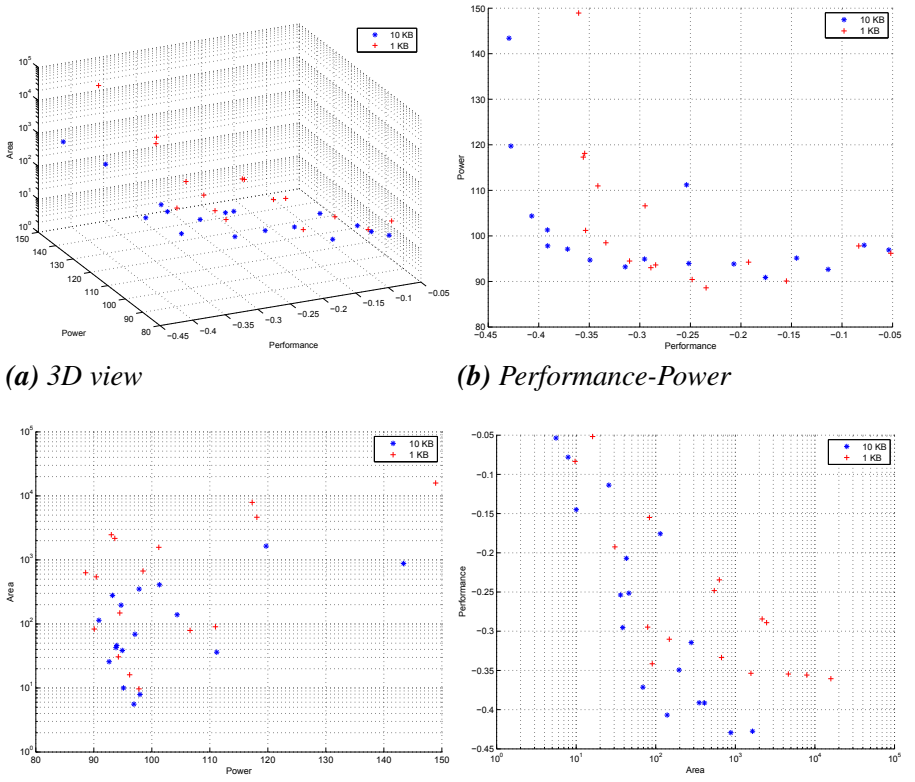
On the contrary, we see slightly bigger area-utilization solutions for the smaller dataset. Although measurements with more dataset sizes are needed to draw safe conclusions, this area trend may again be due to "cold-start" effects; that is, due to the fact that when small datasets are processed, poor CPU utilization occurs since the cache structures do not have enough time to fill, pushing the GA towards larger structures in the hopes of minimizing cache stalls.

6.1.3.2 Framework expansion

As this is one of the first steps towards designing an implant processor, we wanted to make sure that the framework is expandable, in order to facilitate the addition of new domain specific information into the framework as it gets available. In order to test this property of the framework, we devised a synthetic implant application with a hard realtime deadline. Indeed, many implant applications have realtime requirements so formulating a synthetic problem with such a constraint does not fall far from practice. For a demanding, synthetic application we have started from the SiMS case study we have presented in Section 4.7 and added the Motion benchmark to it, as well. In order to also introduce realtime constraints, we have further modified the DMU and Motion benchmarks (called StressDMU and StressMotion, respectively[5]), to represent a single iteration of these (repetitive) applications. This one iteration was chosen to be the *worst-case* iteration in terms of instruction-cycle count for each of the two original benchmarks.

As illustrated in Figure 6.6, the two stress-benchmarks (or *stressmarks*, for short) – combined with data-integrity checks, compression and encryption – represent an *atomic* action for an implant application. In a real-world scenario, this atomic action must be completed, for instance, before the next set of input arrives. Therefore, we *constrain* the total time required for this combined operation as a hard realtime deadline.

[5]These and other modifications in the ImpBench benchmarks will be detailed later, in Section 6.2.



(a) *3D view*

(b) *Performance-Power*

(c) *Power-Area*

(d) *Performance-Area*

Figure 6.5: *Baseline DSE results for* `1 KB` *and* `10 KB` *datasets running on all benchmarks.*

Out of the ImpBench benchmark set, we chose checksum, miniLZO and RC6 as the data-integrity, compression and encryption algorithms, respectively. We obtain the simulated execution time of the processor configuration under investigation from the simulator output. In case the deadline is violated, the processor configuration is deemed to be unacceptable. On the other hand, if the deadline is met, we calculate the objectives of the configuration by combining with the rest of the benchmarks in the test suite (including the original versions of DMU and Motion). Therefore, the fitness metric remains the same as the baseline case.

Figure 6.7 shows the Pareto front evolved with a deadline of `1` second, and also with a slightly relaxed deadline of `2` seconds. As expected, the stricter deadline encourages processor configurations that have a higher performance,

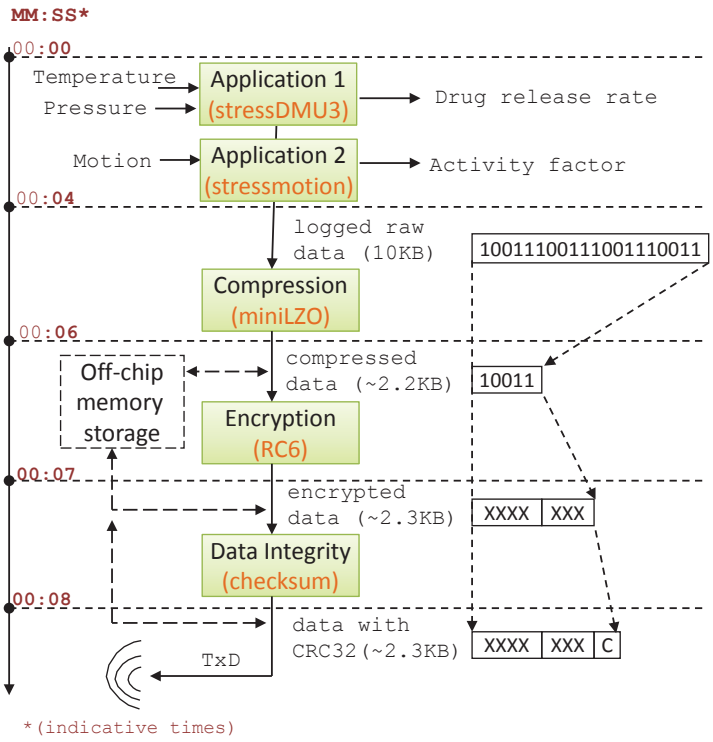


Figure 6.6: *Block diagram of simulated implant application with realtime deadlines.*

but at the same time take slightly more power and area.

6.1.4 Conclusions

In this work we have developed ImpEDE, a novel, multiobjective, framework that provides high-level DSE of biomedical-implant processors, populated by suitable biomedical benchmarks and assorted datasets. ImpEDE organization is described in detail and its functionality is fine-tuned based on our previous experience (e.g. processor parameter values and ranges) and new findings (e.g. crossover probability, dataset size). Restricted by its simulator components, the current framework version can deliver (near) Pareto-optimal processor solutions, co-optimized across performance, power consumption and area utilization. In view of potentially more optimization goals and benchmarks, we have paid attention to making the framework modular and expandable. Furthermore, we have provided a parallelized, versatile version of the

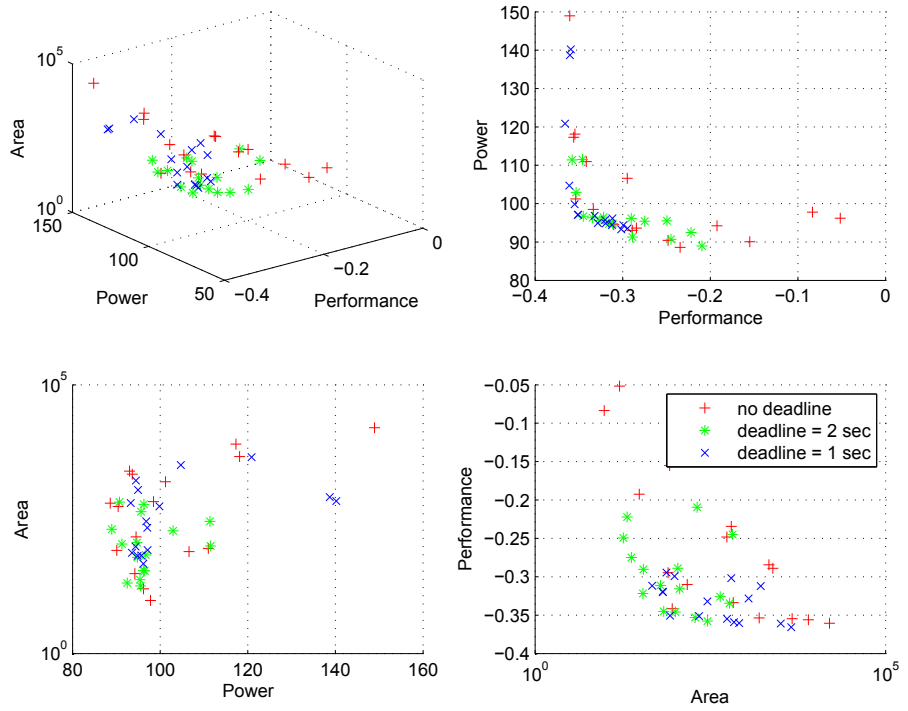


Figure 6.7: *DSE results expanded with hard realtime deadlines of* `2 seconds` *and* `1 second` *for* `10` *KB datasets running on all benchmarks.*

framework which offers execution speedup roughly equal to the number of processor available, without dedicated hardware resources. Last, it has been our intension to make the proposed framework a freely accessible tool, available to everyone online for further studies and improvements.

6.2 ImpBench v1.1: Revisiting the implant benchmark suite

As shown in the previous sections, through the course of developing the ImpEDE framework, we have come across some limitations of ImpBench like, for instance, the absence of any components in the suite for modeling the realtime behavior of a designed processor. For this and some additional reasons, the need for an extended version of ImpBench naturally arose.

In the sections to follow, we attempt to expand the original ImpBench charac-

terization study with extensive, new results acquired through use of ImpEDE: We effectively investigate the *sensitivity* of different processor attributes (e.g. cache geometries) to the various benchmarks in ImpBench. We also update the suite with a new benchmark and two new, so-called "stressmarks", bringing ImpBench to version 1.1. Concisely, this work contributes in:

- Providing a novel and sound methodology, based on GAs, of evaluating benchmark characteristics in terms of resulting processor configurations;
- Identifying a representative (subset of) benchmark(s) for substituting the whole suite in simulations, thus achieving radically shorter DSE times;
- Updating the ImpBench suite to version 1.1 by proposing: (a) a more sophisticated version of the DMU benchmark, and (b) two derived stressmarks for enabling shorter simulation times while biasing the exploration process insignificantly or, at least, predictably; and
- Reporting/amending errata of the original work and giving further clarifications, where needed.

Compared with our own prior work, the new version of the ImpBench suite introduces a more detailed variation of the (originally described) *DMU* benchmark and two stressmarks, that is, two benchmarks based on *DMU* and *motion* and exhibiting worst-case execution (i.e. stress) behavior. A further novelty of the current work is the employment of a GA-based, DSE framework and analytic metrics in order to characterize the old and new ImpBench benchmarks. In effect, this work extends the previous work in both terms of content and methodology. To the best of our knowledge, no benchmark suite has been published before to address the rising family of biomedical-implant processors. What is more, no characterization study has utilized GAs before to explore the benchmark properties and their implications on the targeted processor.

6.2.1 ImpBench v1.1 overview

In order to start our analysis. the original, modified and extended components of the ImpBench benchmark suite are reproduced in Table 6.5. The table further reports binary sizes (built for ARM) and averaged, dynamic instruction/μop counts so as to give a measure of the benchmark complexity. We maintain the original grouping into four distinct categories: *lossless data compression*, *symmetric-key encryption*, *data-integrity* and *real applications*.

benchmark	name	size (KB)	dyn. instr.* (average) (#)	dyn. μops* (average) (#)
Compression	miniLZO	16.30	233,186	323,633
	Finnish	10.40	908,380	2,208,197
Encryption	MISTY1	18.80	1,267,162	2,086,681
	RC6	11.40	863,348	1,272,845
Data integrity	checksum	9.40	62,560	86,211
	CRC32	9.30	418,598	918,872
Real applications	motion	9.44	3,038,032	4,753,084
	DMU4	19.50	36,808,080	43,186,673
	DMU3	19.59	75,344,906	107,301,464
Stressmarks	stressmotion	9.40	288,745	455,855
	stressDMU3	19.52	124,212	224,791

(*) Typical $10 - KB$ datasets have been used, except for DMU-variants which use their own, special datasets.

Table 6.5: *ImpBench v1.1 components and useful general statistics.*

By including groups of different algorithms performing similar functionality in ImpBench, benchmarking diversity has been sought for capturing different processor design aspects. This diversity has already been illustrated in Section 4.6 and will be further elaborated in Section 6.2.3. In this version of ImpBench (v1.1), *real applications* have been expanded with *"DMU3"* and a new category *stressmarks* has been added, featuring *"stressmotion"* and *"stressDMU3"*. The new nomenclature will become clear in the following benchmark descriptions:

MiniLZO (shorthand: *"mlzo"*) is a light-weight subset of the LZO library (LZ77-variant). LZO is a data compression library suitable for data de-/compression in real-time, i.e. it favors speed over compression ratio. LZO is written in ANSI C and is designed to be portable across platforms. MiniLZO implements the LZO1X-1 compressor and both the standard and safe LZO1X decompressor.

Finnish (shorthand: *"fin"*) is a C version of the Finnish submission to the Dr. Dobb's compression contest. It is considered to be one of the fastest DOS compressors and is, in fact, a LZ77-variant, its functionality based on a 2-character memory window.

MISTY1 (shorthand: *"misty"*) is one of the CRYPTREC-recommended 64-bit ciphers and is the predecessor of KASUMI, the 3GPP-endorsed encryption algorithm. MISTY1 is designed for high-speed implementations on hardware as well as software platforms by using only logical operations and table lookups. MISTY1 is a royalty-free, open standard documented in RFC2994 [104] and is considered secure with full 8 rounds.

RC6 (shorthand: *"rc6"*) is a parameterized cipher and has a small code size. RC6 is one of the five finalists that competed in the AES challenge and has reasonable performance. Further, Slijepcevic et al. [126] selected RC6 as the algorithm of choice for WSNs. RC6-32/20/16 with 20 rounds is considered secure.

Checksum (shorthand: *"checksum"*) is an error-detecting code that is mainly used in network protocols (e.g. IP and TCP header checksum). The checksum is calculated by adding the bytes of the data, adding the carry bits to the least significant bytes and then getting the two's complement of the results. The main advantage of the checksum code is that it can be easily implemented using an adder. The main disadvantage is that it cannot detect some types of errors (e.g. reordering the data bytes). In the proposed benchmark, a 16-bit checksum code has been selected which is the most common type used for telecommunications protocols.

CRC32 (shorthand: *"crc32"*) is the Cyclic-Redundancy Check (CRC) is an error-detecting code that is based on polynomial division. The main advantage of the CRC code is its simple implementation in hardware, since the polynomial division can be implemented using a shift register and XOR gates. In the proposed benchmark, the 32-degree polynomial[6] specified in the Ethernet and ATM Adaptation Layer 5 (AAL-5) protocol standards has been selected (same as in NetBench).

Motion (shorthand: *"motion"*) is a kernel based on the algorithm described in the work of Wouters et al. [152]. It is a motion-detection algorithm for the movement of animals. In this algorithm, the degree of activity is actually monitored rather than the exact value of the amplitude of the activity signal. That is, the percentage of samples above a set threshold value in a given monitoring window. In effect, this motion-detection algorithm is a smart, efficient, data-reduction algorithm.

DMU4 (shorthand: *"dmu4"*), formerly known as *DMU*[7], is a real program based on the system described in the work of Cross et al. [25]. It simulates a drug-delivery & monitoring unit (DMU). This program does not and cannot simulate all real-time time aspects of the actual (interrupt-driven) system, such as sensor/actuator-specific control, low-level functionality, transceiver operation and so on. Nonetheless, the emphasis here is on the operations performed

[6] CRC32 generator polynomial: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$.

[7] The original benchmark DMU has been renamed to DMU4, to differentiate it from the new addition DMU3. See main text for further details.

by the implant core in response to external and internal events (i.e. interrupts). A realistic model has been built imitating the real system as closely as possible.

DMU3 (shorthand: *"dmu3"*) is an extension of *"dmu4"*. The original *"dmu4"* benchmark emulates a real-time implantable system by reading real pressure, temperature and integrated-current sensory data (as provided by true field measurements by the implant developers [25]) and by writing to transceiver module (abstracted as a file). *"dmu3"* emulates this in a more sophisticated manner by also accurately modeling the gascell unit used to switch drug delivery in the implant on and off. To this end, it reads additional input data termed "gascell override switch" and "gascell override value". The suffix numbers in both *DMU* benchmarks originate from different field-test runs using different drug-delivery profiles: A high-low-high varying, "bathtub" profile (#4) has been used for *"dmu4"* and a constant, flat profile (#3) has been used for *"dmu3"*. Due to its affinity with *"dmu4"*, *"dmu3"* will not be analyzed further in this work. It has been briefly introduced, though, so that the content of stressmark *"stressdmu3"* will be better understood.

Stressmotion (shorthand: *"stressmotion"*) and **stressDMU3** (shorthand: *"stressdmu3"*) constitute a new addition to the ImpBench suite, their creation stemming from the fact that the original *"motion"* and *"dmu4"* benchmarks have considerably long run times w.r.t. the rest of the benchmarks (see Table 6.5). *"dmu3"* rather than *"dmu4"* has been used for extracting a stressmark due to its more sophisticated emulation of the DMU applications. Since all benchmarks essentially are pieces of continuously iterated code, each stressmark in fact is a derived, worst-case iteration of its respective benchmark. That is, an iteration wherein the implant is required to perform all possible operations; thus, the term "stressmark". As shall be seen in the following analysis, the stressmarks feature significantly shorter execution times.

With the exception of the DMU-variants that use their own internal input data, all other benchmarks come with a full complement of physiological *input datasets* (e.g. EEG, EMG, blood pressure, pulmonary air volume). Without loss of generality, for this work we have, again, selected and executed only the $10 - KB$ EMGII dataset as it exhibits worst-case performance characteristics and, thus, provides a lower-bound for processor design.

6.2.2 Experimental setup

As evaluation framework for our characterization, we have employed ImpEDE. An overview of the framework has already been shown in Figure 6.1.

Optimization (minimization) objectives within the framework are: *processor performance* (in CPI), *processor total area utilization* (in mm^2) and *processor average power consumption* (in mW). As shown in the same figure, a well-known GA (NSGA-II [31]) has been selected for traversing the design space. It generates valid processor configurations also known as "chromosomes". Compromising between unrealistic execution times and quality of results, all full runs of the GA have been allowed to evolve for 200 generations with a population size of 20 chromosomes per generation.

Within the framework, chromosome performance and power metrics are provided by executing the ImpBench benchmarks on the XTREM processor simulator. XTREM settings with ImpEDE have been summarized in Table 6.1. For quantifying each chromosome's area cost, we have used CACTI v3.2, a well-known, cache-area estimation tool.

ImpEDE has primarily been designed for exploring promising implant-processor configurations. However, in this work we employ it as a *meta-tool*, that is, a means of *characterizing* the various ImpBench benchmarks in terms of the different directions they push the GA-based optimization process. In short, we wish to compare the Pareto-optimal fronts of the various ImpBench components and to identify differences in resulting processor configurations.

6.2.3 Benchmark characterization

The first characterization study of ImpBench (see Section 4.6) has focused on illustrating its novelty and variation - thus, its significance - with respect to the most closely related MiBench suite. In the following analysis, we investigate further important attributes within the updated suite. Namely, we address the below questions:

(a) What is the aggregate Pareto front of optimal processor configurations, as driven by the whole ImpBench suite? What are the implications in predicted processor-hardware resources?

(b) What is the contribution to the aggregate Pareto front of each separate benchmark? What is the contribution to the predicted hardware resources?

(c) What is the complexity (in simulation time) of ImpBench as a whole and of its components? With respect to the previous questions, can a representative ImpBench subset with significantly shorter simulation times be identified?

6.2.3.1 Lossless compression

Each complete run of our DSE framework evolving chromosomes for 200 generations, results in - at most - 20 Pareto-optimal, implant-processor configurations. Three-dimensional plots can be produced, revealing the shape of the true[8] Pareto curve. Figure 6.8 illustrates, across the three objectives, three such fronts: the aggregate Pareto front P^{*}[9] labeled in the plots as *"all"* and used as the reference front, and two Pareto fronts formed when only *"mlzo"* and *"fin"* benchmarks are used for the GA evolutions.

Figure 6.8a depicts clear power and performance cut-offs for all three cases with a wide dispersion of chromosomes, indicating a very well-defined design space. We can observe that both compression algorithms achieve a quite distributed performance-power front, yet *"mlzo"* is closer to the aggregate *"all"* than *"fin"* and is, thus, a better candidate for substituting *"all"*. Figure 6.8b reveals that, in terms of area, *"mlzo"* is also closer matching *"all"*, even though it appears to be having slightly higher area requirements.

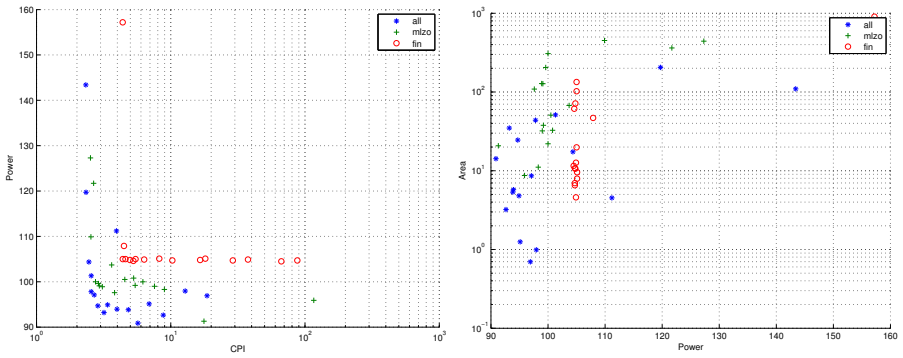
To better understand what these area requirements might translate to in a real processor core, we have put together Figure 6.9. In this Figure, boxplots are drawn for different subsystems of the explored processors. Each boxplot has been created by either running the GA with a single benchmark or the whole ImpBench. Statistics (min, max, median etc.) have been calculated based on the evolved population of 20 processor solutions that reside on the Pareto front. In the current analysis, we will focus on a limited number of observations from this Figure. However, Figure 6.9 contains a large amount of information and can offer the interested reader more predictions on the various processor during architectural exploration.

For the case of lossless-compression benchmarks, Figure 6.9 reveals that *"mlzo"* tends to lead to processors with slightly higher provisions, in particular in the L1 D-cache (D$, hereon) subsystem compared to *"all"* and *"fin"*. For instance, *"mlzo"* requires, on average, a D$ of 64 *KB* size[10] compared to 4 *KB* for *"fin"* and 8 *KB* for *"all"* (see Figures 6.9h, 6.9i and 6.9j). By

[8] In a real-world optimization problem like ours, the true Pareto front is not known. Therefore, we make the reasonable assumption (and have verified to the best of our equipment's capabilities in [27]) that the aggregate front P^{*} reached after 200 generations matches the true Pareto front P,i.e. $|P^{*} - P| \approx 0$. This means that the Pareto front at generation 200 is considered our reference front for comparisons.

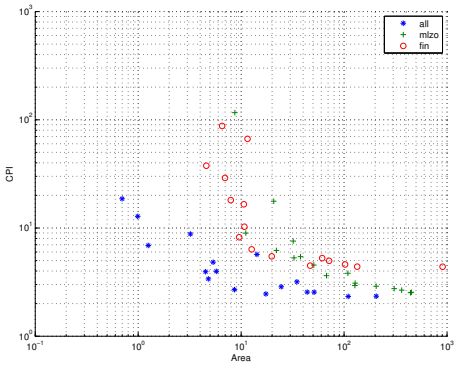
[9] The aggregate Pareto front P^{*} has resulted from running the GA with all original ImpBench benchmarks. That is, *"dmu3"* and both stressmarks are excluded, since they are covered by *"dmu4"*.

[10] It holds that: $cachesize = \#sets * blocksize * associativity$.



(a) *CPI (-) – Power consumption, avg. (mW)*

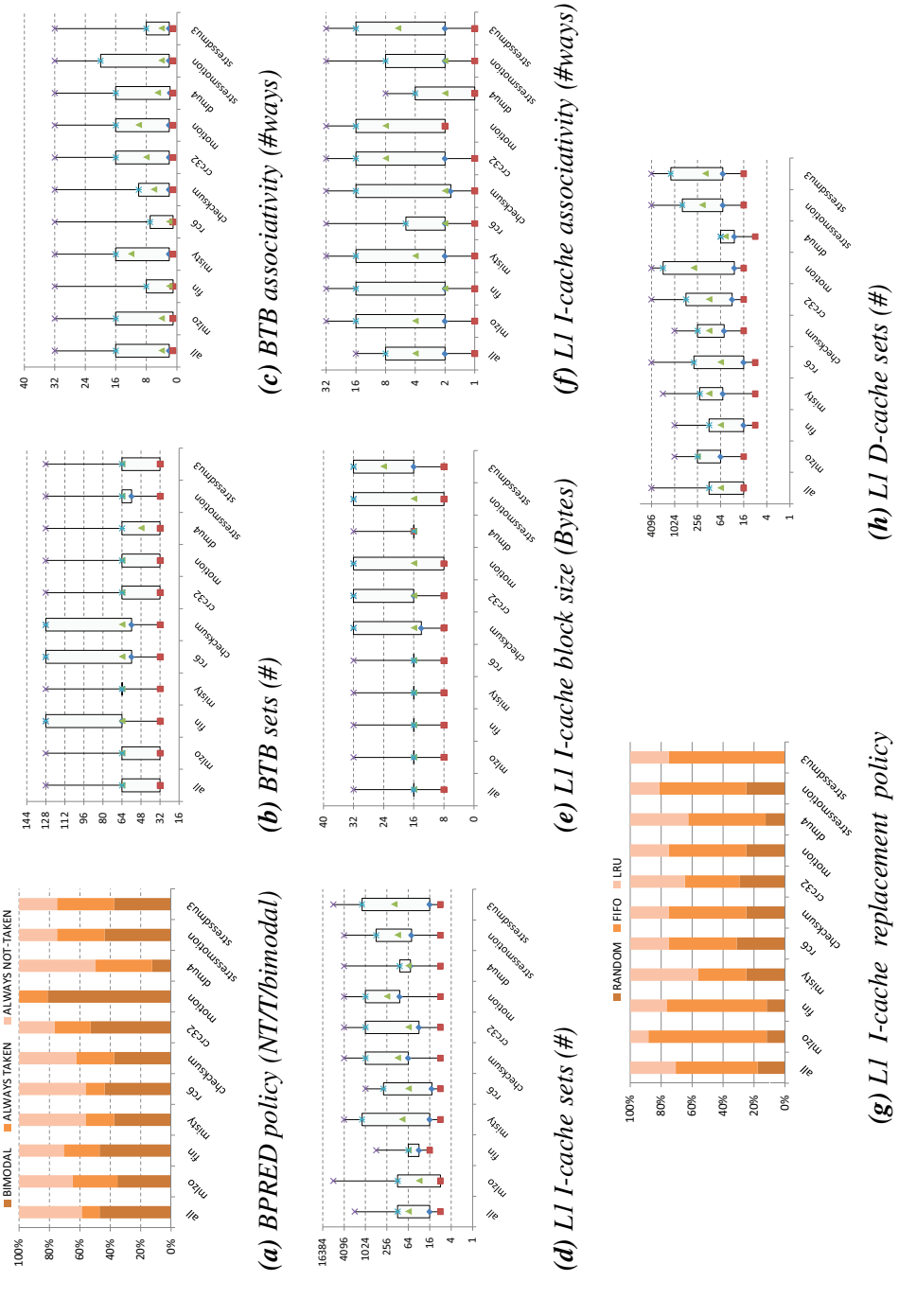
(b) *Power consumption, avg. (mW) – Area (mm^2)*



(c) *Area (mm^2) – CPI (-)*

Figure 6.8: *Final Pareto-fronts (after 200 generations) for lossless-compression benchmarks on $10-KB$ datasets.*

observing the rest of the plots in Figure 6.9, it becomes apparent that, in an overall, *"mlzo"* is very close (slightly worse) to *"all"* in terms of performance and power but - if it substituted the whole ImpBench in the processor DSE - it would lead to more area-hungry processor configurations. Therefore, *"mlzo"* would be an interesting replacement for ImpBench, always giving worst-case design boundaries. *"fin"*, on the other hand, is more obscure in this respect requiring, on average, smaller D$ but larger BTB structures than *"all"*.

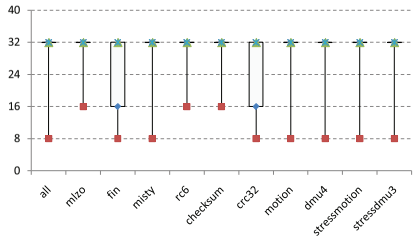


(a) BPRED policy (NT/T/bimodal) *(b) BTB sets (#)* *(c) BTB associativity (#ways)*

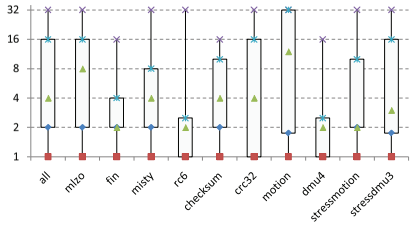
(d) L1 I-cache sets (#) *(e) L1 I-cache block size (Bytes)* *(f) L1 I-cache associativity (#ways)*

(g) L1 I-cache replacement policy (LRU/FIFO/random) *(h) L1 D-cache sets (#)*

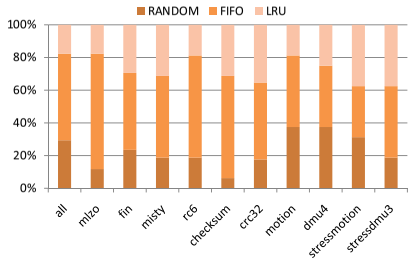
Figure 6.9



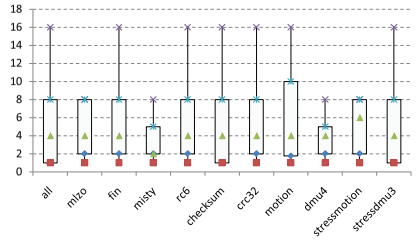
(i) *L1 D-cache block size (Bytes)*



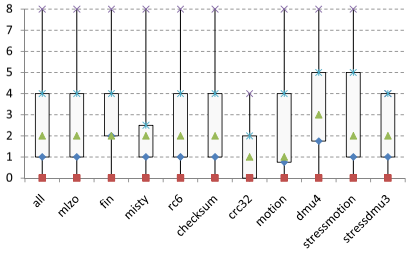
(j) *L1 D-cache associativity (#ways)*



(k) *L1 D-cache replacement policy (LRU/FIFO/random)*



(l) *L1 D-cache miss latency (#c.cycles)*



(m) *Return-address stack (RAS) (#entries)*

Figure 6.9: *Hardware requirements of optimal processors, as evolved over 200-generation runs. Each boxplot is a separate GA run either for a single benchmark or for the whole ImpBench suite and contains statistical results of 20 processors, at most. Barchars are plotted for (a), (g) and (k) since data in these cases are non-numerical. Legend: Medians (triangle), 1st and 3rd quartiles (star and rhombus), min (square), max (x-symbol).*

Benchmark	GD	Δ **(normalized)**	**Sim. time* (average) (sec)**
miniLZO	0.082	0.383	3.07
Finnish	0.115	0.367	21.82
MISTY1	0.083	0.181	16.65
RC6	0.049	0.151	9.37
checksum	0.090	0.189	0.80
CRC32	0.111	0.285	5.46
motion	0.094	0.163	31.16
DMU4	0.085	0.174	37.25
stressmotion	0.088	0.266	1.33
stressDMU3	0.105	0.143	0.60
all	**0.000**	**0.000**	**125.58**

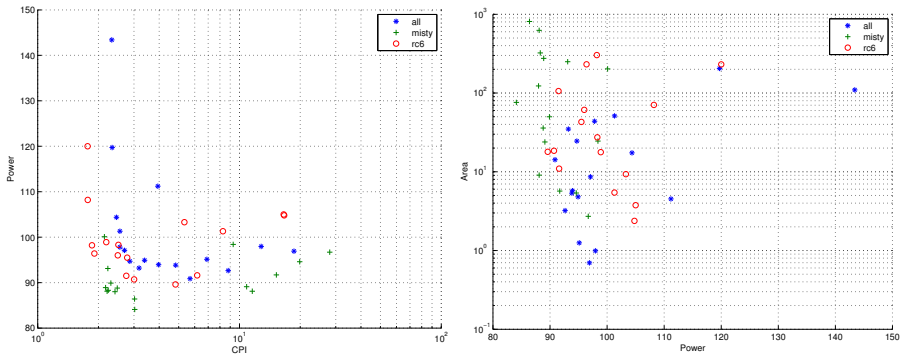
(*) Measured on a dual-core, AMD Athlon(TM) XP 2400+ @ 2000.244 MHz, cache size 256 KB running Fedora 8 Linux.

Table 6.6: *Pareto-front distance and normalized-spread metrics, and average simulation time per benchmark.*

All Pareto fronts Q evolved from single-benchmark runs, present similar (but not identical) 3D-plots. Since results are numerous, we have chosen to also use *arithmetic metrics* to evaluate the benchmarks in a more quantitative and, thus, more reliable manner. For quantifying the **distance** between each single-benchmark Pareto front Q and the reference (aggregate) Pareto front P^*, we have chosen to use Veldhuizen's Generational-Distance GD, based on the previously discussed formula (6.1). Likewise, for quantifying **diversity**, we have used Deb et. al's spread metric Δ, according to the formula (6.2).

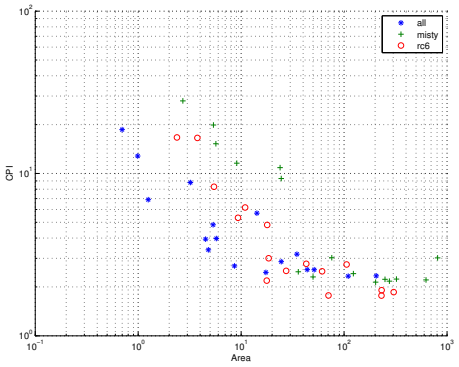
Distance and spread calculations for each benchmark have been accumulated in Table 6.6. Numbers verify the visual observations we have made based on Figure 6.8. The sharper matching of *"mlzo"* to *"all"*, as compared with *"fin"*, is revealed here by the distances of the two compression benchmarks; `0.082` and `0.115`, respectively. In fact, *"mlzo"* features the second smallest GD to *"all"* after *"rc6"*, to be discussed next.

The spread, Δ, of the compression benchmarks, though, is the worst across ImpBench and is slightly better for *"fin"* than it is for *"mlzo"* (`0.367` and `0.383`, respectively), which is especially discernible in Figs. 6.8b and 6.8c. Wider spreads (i.e. smaller Δ's) should imply a wider choice of (optimal) processor configurations from which to pick, as shown in Figure 6.9. From the same Figure we can see that, for the two compression algorithms, BTB sets and



(a) *CPI (-) – Power consumption, avg. (mW)*

(b) *Power consumption, avg. (mW) – Area (mm²)*



(c) *Area (mm²) – CPI (-)*

Figure 6.10: *Final (after 200 generations) Pareto-fronts for symmetric-encryption benchmarks on* $10 - KB$ *datasets.*

BTB associativity vary inversely, resulting in the same overall range of BTB sizes. However, as boxplots in the Figure reveal, *"fin"* displays significantly wider ranges in D$ sizes than *"mlzo"*. This difference could account for the difference in Δ's since all other range differences between the two benchmarks are small.

6.2.3.2 Symmetric encryption

In Figure 6.10 are plotted aggregate, *"misty"* and *"rc6"* Pareto fronts. Although results are more "noisy" than in the case of the compression bench-

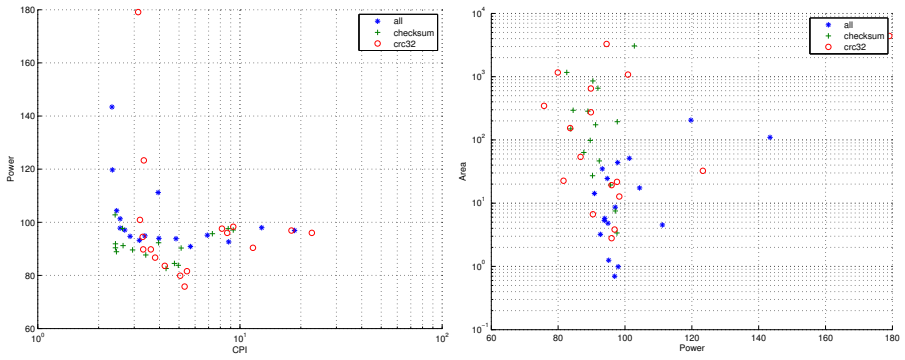
marks, it is clear that both encryption benchmarks are closer to *"all"*, with *"rc6"* practically being on top of it. This is supported by the `GD` values in Table 6.6 which reveals that, in fact *"rc6"* and *"misty"* respectively display the 1st and 3rd smallest `GD`'s overall. Inspection of all three plots of Figure 6.10 further reveals that *"misty"* consumes less power but requires more area than *"rc6"* (and, thus, *"all"*) while scoring better performance than either of them. This agrees with existing literature [134] and with the targeted applications of MISTY1 which are low-power, embedded systems. Its increased area requirements w.r.t. *"all"* are manifest in Table 6.9 across the set and associativity (median) sizes of both the I$ and D$ caches. On the other hand, *"rc6"* features slightly reduced area w.r.t. *"all"*, mainly due to a lower (median) associativity degree in both cache structures.

In terms of diversity of solutions, *"misty"* is somewhat more clustered than *"rc6"* (see Figure 6.10a) and expectedly has a somewhat larger Δ. Yet, both benchmarks display good spreads with *"rc6"* ranking overall 2$^{\text{nd}}$ best. This essentially means that a number of diverse (optimal) processor configurations exists for servicing either benchmark. In particular *"rc6"* is a highly scalable application, which can lead to diverse processors while maintaining low area requirements (as seen above). To be precise, Figure 6.9 reveals that *"rc6"* leads to processors with at least $\times 4$ smaller BTB, I$ and D$ structures without sacrificing processor flexibility. For instance, the D$-set boxplot (Figure 6.9h) of *"rc6"* indicates most popular sizes from 16 to 64 entries.

In terms of D$ miss latency, *"rc6"* is also more relaxed compared to *"misty"*. It allows for latencies up to `8 cycles` (which is similar to what the aggregate run indicates) while *"misty"* can tolerate maximum miss penalties of `5 cycles` (median). That *"misty"* can lead to overestimations in latency (and overall performance) can also be clearly seen by its increased CPI w.r.t. *"all"* in Figure 6.10c. To sum up, for the case of the encryption benchmarks, if *"misty"* was selected alone to drive the exploration process, it would underestimate performance and power costs and overestimate area costs.

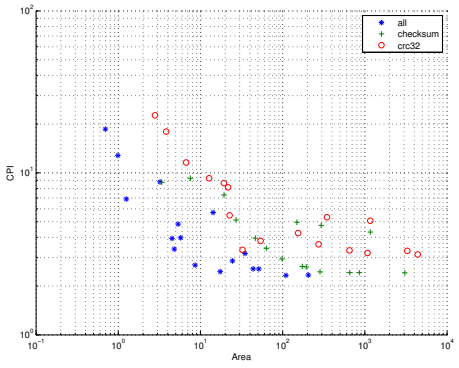
6.2.3.3 Data integrity

Figure 6.11 illustrates *"checksum"* and *"crc32"* Pareto fronts, along with aggregate, *"all"* Pareto fronts. Compared to *"all"*, *"checksum"* is somewhat slower and requires more area. *"crc32"* is even slower and results in processor-area costs $\times 2$ those of *"all"*. As an indication of the different area ranges involved, from Figs. 6.9b and 6.9c, *"all"* leads the exploration to a median BTB size of `2 KB` while *"checksum"* to `3 KB` and *"crc32"* to `4 KB`. These observa-



(a) *CPI (-) – Power consumption, avg. (mW)*

(b) *Power consumption, avg. (mW) – Area (mm^2)*



(c) *Area (mm^2) – CPI (-)*

Figure 6.11: *Final (after 200 generations) Pareto-fronts for data-integrity benchmarks on $10-KB$ datasets.*

tions are corroborated by the GD of *"checksum"* being equal to 0.090 and that of *"crc32"* being worse and equal to 0.111.

"crc32" is also more irregularly distributed than *"checksum"*, therefore its Δ (0.285) is much worse than that of *"checksum"* (0.189) which is following *"all"* more closely. Similarly to *"rc6"*, *"checksum"* offers a wide spectrum of processor alternatives, while *"crc32"* results in more clustered solutions. It should be noted that this clustering is not always a downside of a benchmark, especially in cases where we are interested in neighboring alternatives in the same design niche. It offers finer resolution within the area of interest.

The proximity of *"checksum"* to *"all"* is also reflected in Figure 6.9 on the

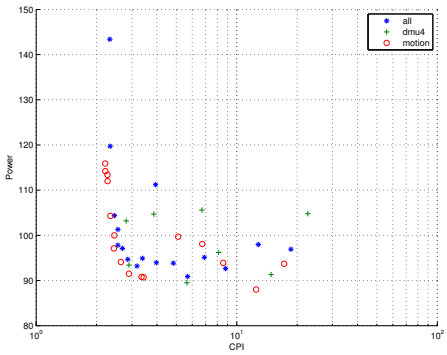
identical boxplots for the D$ miss latency and the RAS size. Yet, *"checksum"*'s simpler structure seems to favor the simpler branch-prediction technique ALWAYS TAKEN while the more complex *"crc32"* tends towards the more powerful BIMODAL predictor. Overall, *"checksum"* appears capable of substituting *"all"* in simulations but we should always account for the observed increase in area and decrease in performance of the resulting processor configurations. *"crc32"*, on the other hand, features the second worst GD after *"fin"* and its Δ is mediocre. Combined with its high area costs, it should not be considered in most cases as a good, single substitute for *"all"*.

6.2.3.4 Real applications & stressmarks

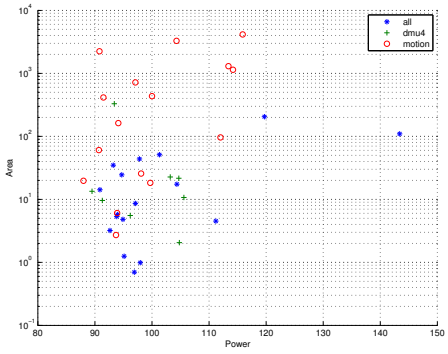
In Figure 6.12, Pareto fronts for the real applications *"motion"* and *"dmu4"* are compared with *"all"*. The plots reveal good fitting and dispersal of the solutions for both benchmarks, and GD/Δ figures agree with these observations. Power and performance ranges are similar to *"all"*, yet area ranges are significantly different. Although *"motion"* has a much simpler functionality than *"dmu4"*, it incurs disproportional area costs: on average $32\ KB$ for the BPRED/BTB, $120\ KB$ for the D$ and $5\ KB$ for the I$, as opposed to *"dmu4"* which promotes processors with small average sizes: $2\ KB$, $3\ KB$ and $1.8\ KB$, respectively.

Regarding branch-prediction hardware, *"motion"* presents unexpected results, too. It almost exclusively favors configurations equipped with a BIMODAL branch predictor. Conversely, *"dmu4"* opts mainly for the simpler static predictors ALWAYS TAKEN and ALWAYS NOT-TAKEN. On the other hand, *"dmu4"* appears to suffer more from increased D$ miss latencies, compared to the average case. In effect, *"all"* evolves processor configurations with latencies up to 8 cycles while *"motion"* drives latencies up to 10 cycles and *"dmu4"* only up to 5 cycles. As opposed to the preceding benchmarks, in this case, *"motion"* and *"dmu4"* display diverse properties which cannot be covered fully by either single one of them. This is to be expected for benchmarks (in essence, kernels) emulating implant applications. If one real benchmark had to be selected as representative for this group, *"dmu4"* would be the safer choice.

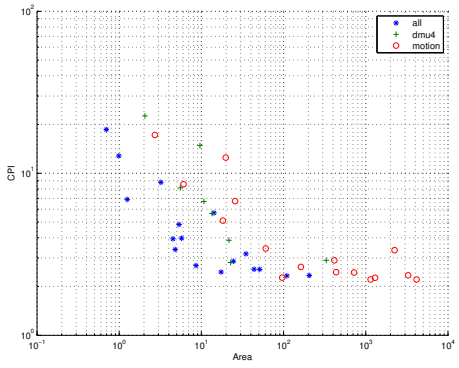
In order to compare the characteristics of the real benchmarks, above, and the newly-created stressmarks, we have plotted Figure 6.13, where the Pareto fronts of all four programs are being shown. Plot 6.13a reveals that, in terms of performance and power, *"stressmotion"* has a close distance to *"motion"* albeit a slightly worse spread. In terms of hardware requirements, *"stress-*



(a) *CPI (-) – Power consumption, avg. (mW)*



(b) *Power consumption, avg. (mW) – Area (mm^2)*

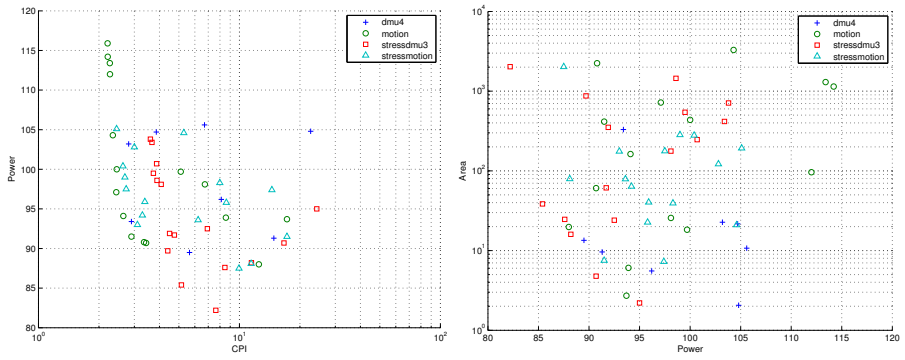


(c) *Area (mm^2) – CPI (-)*

Figure 6.12: *Final (after 200 generations) Pareto-fronts for real applications on* $10 - KB$ *datasets.*

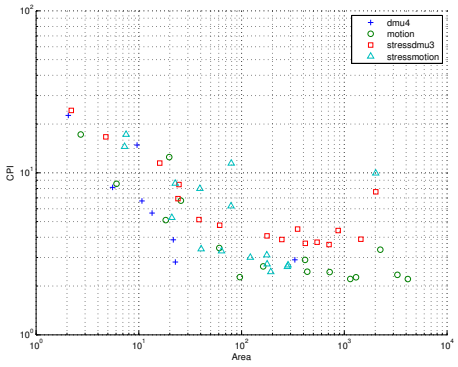
motion" relaxes design requirements compared to *"motion"*: BPRED/BTB 4 *KB*, D$ 12 *KB* and I$ 2 *KB* on average. In an overall, though, by combining *"stressmotion"* from Figure 6.13 with *"all"* from Figure 6.12, we can see that *"stressmotion"* is actually closer to the aggregate Pareto line than *"motion"*, as verified by the respective *GD*'s. Essentially, *"stressmotion"* can track the Pareto front better than the full *"motion"* benchmark, albeit with somewhat more clustered solutions.

"stressdmu3", on the other hand, follows *"all"* with somewhat less fidelity than *"dmu4"* but displays a better spread. As Figure 6.13 indicates, both *"stressdmu3"* and *"dmu4"* fronts are residing in the same locus of solutions.



(a) *CPI (-) – Power consumption, avg. (mW)*

(b) *Power consumption, avg. (mW) – Area (mm²)*



(c) *Area (mm²) – CPI (-)*

Figure 6.13: *Final (after 200 generations) Pareto-fronts for real applications and stressmarks on their respective datasets.*

Yet, *"stressdmu3"* achieves slightly lower performance and drives processor resources slightly up compared to *"dmu4"*, as follows: BPRED/BTB 22.5 *KB*, D$ 15 *KB* and I$ 2 *KB* on average.

As far as branch-prediction requirements are concerned, GA evolutions reveal the following: Since stressmarks run for a short period of time (one or a few iterations only), branch-predictor distributions in the barcharts of Figure 6.9a are relatively unaffected. The simpler scheme ALWAYS TAKEN is mostly expanded but, other than that, both stressmarks bear distributions similar to *"all"*.

Conclusively, the new stressmarks track the true Pareto front closely, each either scoring better in *GD* or Δ. It is interesting to notice that, while they have

not contributed to the true Pareto front (i.e. they have not been used in the aggregated-benchmark simulations), the solutions they evolve are highly comparable and in the same locus as the aggregate solutions. This fact attests to their prediction quality and, provided that attention is paid to the differences they exhibit (as discussed above), they can offer reliable substitutes for the full real applications.

6.2.4 Conclusions

The previous analysis has unveiled new information with respect to the coverage of the design space and the hardware implications contributed by each ImpBench component. In this context, results indicate that, from the lossless-compression benchmarks, *"mlzo"* provides better GD and similar Δ to *"fin"*. On top of this, it also features faster simulation times (about $\times 3$ faster, according to Table 6.6). From the symmetric-encryption benchmarks, *"rc6"* displays excellent characteristics, namely the smallest GD and the second best Δ, meaning that it traces the aggregate Pareto front with high fidelity. According to Table 6.6, it is also about double as fast as *"misty"*. Of the data-integrity benchmarks, *"checksum"* performs consistently better than *"crc32"* and features the overall shortest simulation time ($0.80\ sec$) across all full benchmarks.

For the real-applications case, no single benchmark could be unanimously ranked higher due, mainly, to the complex nature of the results. This actually shows the usefulness of both real benchmarks which, also, feature similar simulation times (and the largest in the whole suite). Last, analysis of the new stressmarks has revealed that, although they display some variability in predicted processor specifications w.r.t. the full real applications, they both track the true Pareto front closely. Careful use of the stressmarks can seriously reduce simulation times up to $\times 30$ (see Table 6.6), which is an impressive speedup and a good tradeoff between DSE speed and accuracy.

The above results indicate highest-ranking benchmarks within the ImpBench suite, however this is not to say that the poorest-performing ones are redundant. The findings of the original analysis (in Section 4.6) indicate that each benchmark in ImpBench exhibits diverse characteristics (e.g. μop mixes) and should not be dropped from consideration when considering implant-processor design. On the contrary, this study comes as a complement and extension of the original ImpBench study.

6.3 Exploration of optimal SiMS Processors

In this third and last part of the current chapter, we finally put ImpEDE to the use that it was originally intended for: We present the results of an automated, DSE effort performed to identify optimal SiMS-processor candidates. We also select a number of representative, real implant applications in the literature and explore the possibility of covering them with a few of the identified processors. Concisely, the contributions of this work are:

- To propose a new, realistic, worst-case workload mix for future implant processors;
- Along with the previously generated DSE toolset and the new workload mix, to provide a complete framework enabling the implant designer to make informed decisions about resource allocation for future implant design;
- To propose Pareto-optimal, alternative microarchitectural configurations for the SiMS processor;
- To make a proof-of-concept, first attempt at fitting a single (or a few) of the identified configurations to real implant applications.

It should be noted that this study focuses on the microarchitectural aspects of the SiMS processor, thus no Instruction-Set-Architecture (ISA) analysis is present. The work presented here is original in that it attempts to propose a generic and low-power processor architecture while at the same time providing the performance needed by current and future applications in the field. A *systematic*, *structured* approach to the problem, supported by the recent, rapid advances in microelectronics technology [66], finally make such a venture realistic.

6.3.1 Experimental setup

6.3.1.1 Exploration framework

In order to perform automated exploration, we have employed ImpEDE; an overview of the framework can been seen in Figure 6.1. Within the framework, performance (i.e. execution time) and power metrics are provided by utilizing XTREM. A summary of the XTREM parameters, as configured for ImpEDE,

is included in Table 6.1. More advanced microarchitectural structures such as caches and branch predictors have not been disabled in XTREM as they have been shown (in Chapter 5) to be relevant within the implant context.

While XTREM has been very useful in our studies so far, it is not an ideal simulator. One of the major drawbacks of using XTREM is that it models a low-power, high-performance embedded processor – an overkill in the implant application domain. Another shortcoming is that it does not simulate any (off-chip) memory, thus making system-level simulations difficult. Also, our long usage of XTREM has revealed a number of bugs and modeling inaccuracies (see [28] for an extensive list), most of which have been solved by a newer simulator XEEMU [60]. Therefore, an XEEMU porting for our framework is in the process of being developed. In the meantime, XTREM has been maintained in our exploration chiefly for reasons of compatibility with previous work, availability and ease of use. We have combined readouts from XEEMU regarding memory power consumption and have updated the power metric in our exploration in order to overcome some of the XTREM limitations.

For quantifying each chromosome's area cost, we have used CACTI v3.2, a well-known, cache-area estimation tool. The total area cost has been calculated as the sum of the (fixed) net processor and (off-chip) memory area, based on related literature; and the per-case cache (BTB, I$, D$ etc.) estimates derived from CACTI simulations.

6.3.1.2 Worst-case workload mix

In order to drive the exploration process, the employed ImpEDE framework has been supplemented by ImpBench-v1.1 components which are able to capture both the *functional* as well as *timing* behavior of profiled implant processors (see Table 6.5).

Given that implants constitute mission-critical devices and, to dull the effect of any accuracy errors introduced in this DSE study, we are interested in identifying and using a *worst-case workload mix* that will characterize future implant processors. Such a mix has already been presented in Section 6.1.3.1. For convenience, this mix is drawn in Figure 6.14 once more.

The rationale of synthesizing this particular implant application is as follows: In order to provide a realistic, *worst-case*, SiMS-processor design, we have selected, per ImpBench-v1.1 benchmark category, the fastest executing algorithm; that is, *miniLZO* for compression, *RC6* for encryption and *checksum* for data integrity (refer to Table 6.5 for the various execution times). As a

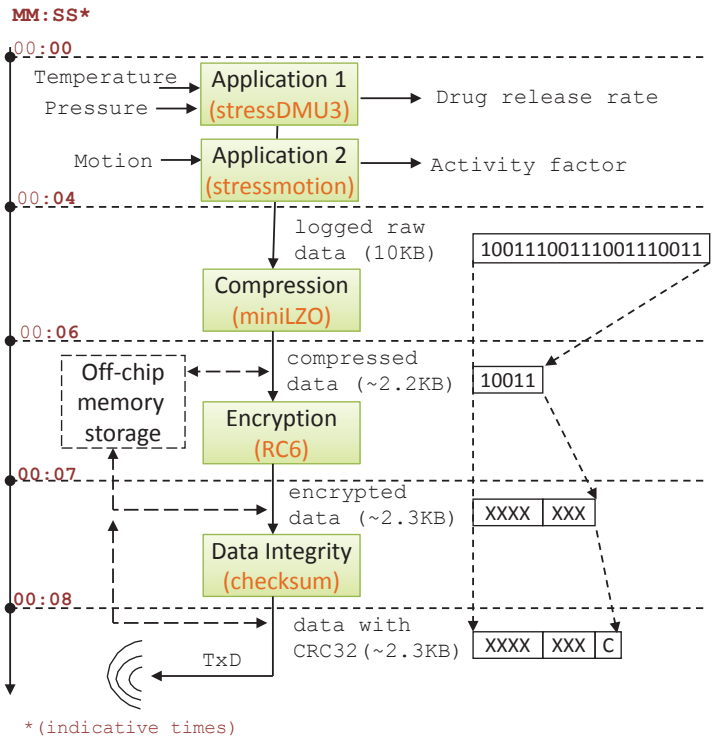


Figure 6.14: *Block diagram of simulated implant application with realtime deadlines.*

real-application benchmark, we have chosen both stressmarks *stressmotion* and *stressDMU3*, which simulate a single-iteration, worst-case instance of the regular benchmarks *motion* and *DMU3*, respectively. In combining the above benchmarks in the mix, we have attempted to include the heaviest processing tasks to be executed at the narrowest window possible.

Every processor configuration (or chromosome) evolved through ImpEDE is made to execute this whole sequence of benchmarks, representing *the busiest (i.e. worst-case) iteration* in the implant's operational lifetime. The *execution-time* metric is calculated as the accumulation of execution times of all involved benchmarks while the *power-consumption* metric is calculated as the weighted average of the power consumptions of all involved benchmarks with each one's execution time used as the weighting coefficient.

To push the worst-case, processor-design envelope further, and without loss of generality, we use $10 - KB$ EMGII as the input dataset to the above bench-

marks. It features a realistic size and has been shown to evoke the longest execution times among the available physiological datasets [27].

It should be noted, last, that all ImpBench benchmarks (and, thus, the ones currently used) are kernels simulating the processing load of an implant processor. Therefore, they suffer from certain modeling limitations: they have no way (a) of modeling the behavior of any implant peripherals (biosensors/bioactuators), and subsequently (b) of accurate modeling any externally triggered (timing or other) events, i.e. they have no sense of real time. This is a well-known problem in benchmarking (event-driven) embedded systems. This has been addressed by introducing extra code in the benchmarks to imitate the passage of time and the occurrence of external events (e.g. timer/sensor interrupt). This, of course, has to be done in a careful fashion as it can potentially pollute simulation results in terms of timing behavior, executed instruction mix and so on.

6.3.2 SiMS-processor DSE execution

With the above considerations, ImpEDE has been allowed to run over significant periods of time in search of optimal SiMS-processor configurations. Table 6.7 lists the results of this search. Each one of the 19 entries is a Pareto-optimal, non-dominated solution to the problem. Performance, power and area metrics are also reported for each entry.

6.3.3 Implant study cases

For selecting representative study cases of the implant application domain, we draw upon the study cases of the survey performed in Chapter 2. The selected applications will help provide diverse operational requirements for our targeted SiMS processor(s).

cnf	BPRED	BTB sets	BTB assc	RAS	L1-I$ sets	L1-I$ bl.size	L1-I$ assc	L1-I$ repl	L1-D$ sets	L1-D$ bl.size	L1-D$ assc	L1-D$ repl	Mem lat	Ex. Time	Power	Area
	(-)	(#)	(#)	(#)	(#)	(bits)	(#)	(-)	(#)	(bits)	(#)	(-)	(#cc)	(sec)	(mW)	(mm^2)
1	bimod	64	8	8	4096	16	32	FIFO	4096	16	1	FIFO	2	27.465	17.539	2521.36
2	bimod	128	8	2	256	16	16	LRU	4096	8	2	LRU	16	37.166	15.368	394.53
3	bimod	64	32	0	256	16	16	RAND	1024	32	2	FIFO	1	1.790	123.143	400.92
5	taken			1	1024	32	32	RAND	16	16	16	FIFO	8	26.143	13.842	1325.39
6	nottaken			8	1024	16	4	FIFO	512	32	2	FIFO	1	1.433	63.217	327.10
7	bimod	128	8	4	2048	16	8	FIFO	4096	32	1	LRU	1	1.751	93.200	659.94
8	taken			0	16	32	8	RAND	512	32	4	RAND	8	2.777	74.860	299.37
9	bimod	128	2	4	64	32	8	LRU	128	32	16	FIFO	8	2.181	63.288	327.61
10	bimod	32	16	8	128	8	2	FIFO	16	32	8	RAND	1	4.516	87.887	243.68
11	bimod	64	8	1	256	16	4	FIFO	64	32	16	FIFO	2	1.951	93.366	298.79
12	nottaken			4	16	8	2	FIFO	64	8	2	RAND	8	35.571	88.153	215.30
13	bimod	64	4	2	8	16	1	FIFO	128	32	2	RAND	16	6.834	69.729	227.71
14	nottaken			2	64	16	2	LRU	16	32	16	FIFO	16	4.605	67.197	250.99
15	nottaken			2	16	8	4	FIFO	32	32	2	FIFO	4	6.823	80.947	218.21
16	nottaken			2	8	32	2	LRU	64	16	2	FIFO	1	24.463	71.681	218.84
17	nottaken			1	32	16	16	FIFO	16	32	4	LRU	8	2.868	69.781	238.62
18	bimod	128	2	8	64	8	16	FIFO	16	32	16	FIFO	2	2.222	90.816	268.36
19	bimod	32	1	8	64	16	16	LRU	128	32	32	FIFO	4	1.922	74.419	421.30
20	nottaken			1	128	16	4	LRU	64	32	4	RAND	16	3.395	62.336	236.57

Table 6.7: *ImpEDE-evolved, optimal processor configurations.*

In order for a direct and fair comparison with the candidate SiMS processor(s), we have to place the study cases in the same design space as the one traversed by ImpEDE. That is, we need to know the worst-case execution time, the power consumption and the area cost of each of the studied implantable systems. This requirement limits the number of eligible systems to only 6, as shown in Table 6.8. In spite of this, the scope of applications addressed is diverse – spanning the muscular, neural, cardiac, gastric, atrial, and nervous systems. An extensive description of the various devices can be found in [132], yet short descriptions are given below for convenience.

Device #A, by Smith et al. [111, 112, 127], is used for functional neuromuscular stimulation (FNS). The authors are describing a flexible implantable-stimulator and telemetry (IST) system which makes provisions for multiple channels of stimulation, multiple channels of sensor or biopotential-electrode sensing and power and bidirectional data communication between the implant and an external control unit (ECU) over a transcutaneous, inductive RF link.

Device #B, by Eggers et al. [36, 37, 62], is a miniature, implantable, intracranial pressure (ICP) measurement system for monitoring patients in the ER (e.g. post-surgery patients). This essentially is a telemetry-powered, implantable system consisting of an absolute-pressure sensor and two low-power ASICs for pressure read-out and telemetric data/power transmission.

Rollins et al. [115] have developed an implantable radio-telemetry system (**device #C**) for continuous monitoring of ECG signals over a period of weeks to months for capturing all events preceding sudden-death incidents. The design of the system centers around two separate but inter-dependent units: the implantable unit and a backpack which holds batteries for powering the implant, a processor and a WLAN-card for forwarding the data wirelessly to a base station for further archiving and analysis.

Valdastri et al. [140] present a new, versatile implantable system (**device #D**) that provides multichannel telemetry of measured biosignals. The presented system consists of the microcontroller-based implant which can monitor and wirelessly transmit up to 3 channels to an external receiver and, in this case, monitors gastric pressure in the stomach.

Au-Yeung et al. [7] have built an implantable **device (#E)** which is capable of continuously monitoring the electrophysiological state of the heart atria and, also, of delivering chronic and programmable atrial pacing. In effect, the proposed system can induce standard AF, can measure the atrial effective refractory period (AERP), can deliver anti-tachycardia pacing (ATP) therapy and can sense and telemeter atrial electrograms (AEGs).

case	Author	Pub. Year	Application	Power source (-)	Sensor count (#)	Sampl. rate (Hz)	ADC resol. (bits)	Core arch. (-)	Core freq. (MHz)	**Ex.Time Worst-case (sec)**	**Power Peak (mW)**	**c/s Area Total (mm^2)**
A	Smith et al. [111, 112, 127]	1998	restoration of paralyzed muscle, MES	RF-ind.	2	100	12	FSM	1	34.1333	96.00	937.50
B	Eggers et al. [36, 37, 62]	2000	ICP-based diagnosis for brain diseases	RF-ind.	1	100	10	no	0.125	81.9200	0.24	58.50
C	Rollins et al. [115]	2000	continuous ECG for spontaneous cardiac arrhythmias	battery (ext.)	8	1000	12	FSM	2	0.8533	34.00	4209.67
D	Valdastri et al. [140]	2004	gastric-pressure monitoring	battery	1	25000	10	8-bit μC	4	0.3277	50.40	162.00
E	Au-Yeung et al. [7]	2004	continuous AEG, delivery of atrial ATP	battery	4	333	10	8-bit μC	8	6.1502	115.30	5106.00
F	Liang et al. [84]	2005	ENG	RF-ind.	1	11000	10	8-bit μC	n/a	0.7447	90.00	1350.00

Table 6.8: *Study cases of real implantable applications (taken from the survey results of Chapter 2).*

The developed system by Liang et al. [84] (**device #F**) allows recording and telemetry of electroneurogram (ENG) signals to an external host computer. The implant is built to receive power and ASK-modulated commands over a wireless RF-link and to transmit physiological data back through passive telemetry. The device consists of a μC with on-chip, $10-bit$ ADC which digitizes and forwards data acquired from an analog-sensing front-end through cuff electrodes.

As illustrated in Table 6.8, actual implant chipset sizes have been employed for the area metric. The term 'chipset' represents the dimensions of any design and assembly type; ranging from fully integrated and multi-chip module (MCM), to PCB-mounted. Figures were also available for the implant chip-only size (in mm^2) - e.g. processor die but no supporting PCB - and for the implant package size (in mm^3). However, the chipset area was finally preferred so as to allow more direct and fair comparisons with the XTREM processor plus off-chip memory. Memory is specifically included in this work as the initial analysis (see Section 2.6.4) revealed rising trends in memory usage for future implants.

As far as power consumption is concerned, the most frequently reported figure in the actual implantable systems is active (peak) power which was the power measured during full load. This is the power simulated by XTREM as well, since XTREM does not support any low-power or sleep modes of operation. Therefore, peak power values as reported by XTREM, without any conversion, have been used as the power metric.

Finally, for the performance metric, some estimations were required in order to make the real and simulated systems commensurable. All case studies are devices with periodic monitoring windows, thus exhibiting a specific sampling rate, as shown in Table 6.8. The inverse of this rate (or frequency) signifies the maximal amount of time the device has to read a sensor value and process it before the next value arrives. In effect, this is the worst-case execution time of the implant. Note that we might have used the 'Core frequency' as a measure of the processing rate but this would be accurate only for designs with very simplistic cores as in cases #A, #B and #C. For the rest of the cases whereby a full μC is used, the core frequency is much higher (typically three orders of magnitude) than the actual sampling frequency and, thus, does not reflect the real-time deadlines of the implant.

However, the performance metric for the study cases (as the inverse of the sampling rate) is not yet completely normalized with respect to that of our processor configurations. As discussed in the previous section, our processor configurations consume EMG input data of $10\ KB$. The study cases, on the

other hand, are assigned (by design) the task of consuming a single sample of size equal to the ADC resolution used (e.g. $8\ bits$), from each sensor they have on-board. Therefore, for each study case to collect $10\ KB$ of sample data, a longer execution time is needed which is inversely proportional to the number of available sensors. The normalized, worst-case execution time is then given by:

$$ET_{norm} = \frac{10\ KByte}{F \times N \times S}, \tag{6.3}$$

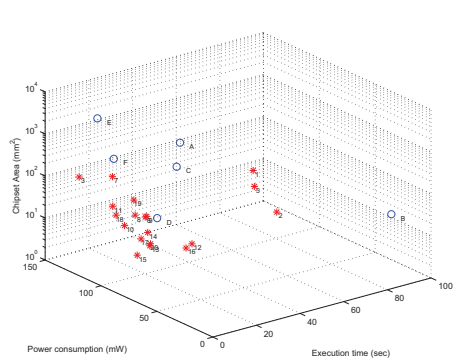
where F is the sampling frequency (in Hz) of the sensor(s), N is the ADC resolution (in bits) and S is the number of working sensors on the implant. This is the execution time given in Table 6.8. It should be noted that formula (6.3) does not account for any further processing of the data once acquired. On the contrary, our evolved processor configurations perform significant processing tasks, as discussed in Section 6.3.1.2. Therefore, by considering an ideal situation involving zero processing time for the real life comparison cases, while keeping a non-zero processing time for our designs, we design our implant processors for the worst-case.

6.3.4 Exploration results

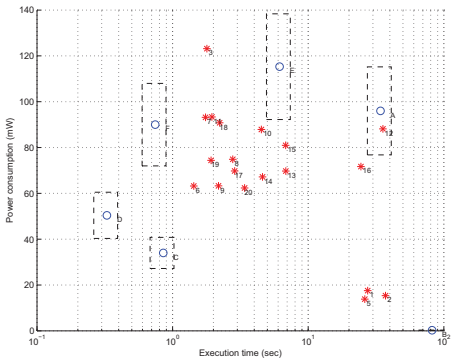
In this section, we see how a single processor or family of processors can be identified as "generic processor(s) for implants". We denote this set of processors as $\mathbb{P}$. Once developed, this family should be able to replace a large number of implant applications - i.e., there must be a processor p in the set $\mathbb{P}$ that has equivalent or better design characteristics than the existing application in question. For reasons of economy, $\mathbb{P}$ must be a minimal set.

As mentioned before, we consider 3 design characteristics – power, performance and area. Therefore, we have a 3D design space, as shown in Figure 6.15a. In this Figure, our processor design points (denoted with numbers) and the case-study points (denoted with letters) have been plotted. For clarity purposes, 2D Figures 6.15b, 6.15c and 6.15d of the same 3D space have also been plotted. The bounding boxes around the study cases represent 10% confidence intervals to compensate for the uncertainty introduced when trying to fit the study cases in the design space.

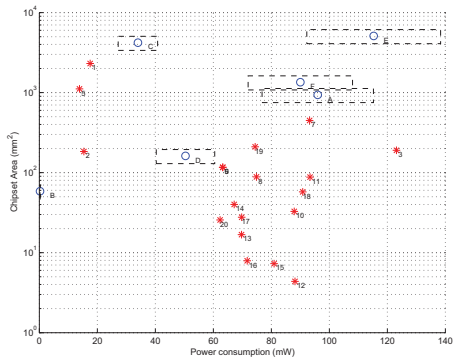
From the figures, we notice that implant devices #A and #E are dominated by most of the candidate processor points in all three dimensions. Therefore we can include any configuration from {*6, 7, 8, 9, 10, 11, 13, 14, 15, 17, 18, 19,*



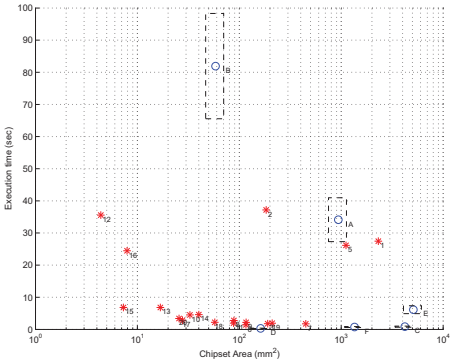
(a) *3D view*



(b) *Worst-case, single-loop execution time - Average power consumption*



(c) *Average power consumption - Total chipset area*



(d) *Total chipset area - Worst-case, single-loop execution time*

Figure 6.15: *Comparison of study cases and DSE results for* `10 KB` *datasets running on the selected benchmarks.*

20}[11] in order to cover applications #A (restoration of paralyzed muscle) and #E (atrial electrogram and anti-tachycardia pacing).

The other devices are not so easily fitted without applying standard engineering practices. For example, device #B, is largely dominated in terms of execution time and area but not in terms of power. In fact, #B has the lowest power profile (`0.24 mW`) by a wide margin across processor configurations and study cases alike. Looking at the application details [37], we see that, #B is a min-

[11] As labeled in Figure 6.15.

imal functionality device measuring intra-cranial pressure and is powered by an *external* power source (through RF induction). Therefore, power is not a big issue for this application as it allows for a non-implanted power source – which in practice can be replaced by a bigger power source if required without compromising the implanted chip area. Therefore, any of the processors that dominate #B across the other two dimensions may still be used to replace it – with the provision of a bigger external power source, and keeping in mind heat dissipation constraints. Therefore, #B (diagnosis of brain disease) can be replaced by $\{10, 12, 13, 14, 15, 16, 17, 18, 20\}$.

Furthermore, we see that devices #C, #D and #F are highly performance oriented. Out of these, device #D performs gastric-pressure monitoring at the extremely high sampling rate of 25 `kHz`, the highest rate among all applications in the study. In practice, however, gastric pressure varies at a much lower rate (making 1-5 second samples more than sufficient), making this a good example of implant over-design. We see that configuration $\{2\}$ dominates case #D if this fact is taken into consideration, and can be a suitable replacement in practice. On the other hand, devices #C and #F perform continuous-ECG and ENG monitoring, which are indeed demanding applications in terms of throughput and, therefore, cannot be accommodated by a lower sampling rate. We observe that these devices are dominated by $\{1, 2, 5\}$ and $\{2, 5\}$ respectively w.r.t. power and area. We see that configuration $\{2\}$ is present in all three replacement sets. Therefore, if $\{2\}$ were available as a cost-effective, generic, pre-tested and pre-approved component as envisioned, application #D can be replaced without loss of functionality; #C, #F could be accommodated by adding a hardware accelerator in order to deliver the required performance. Such a hardware accelerator is feasible as long as it falls in the power and area margin provided by $\{2\}$ as compared to the application in question.

6.3.5 Discussion

From the above analysis, we can make the following observations: First off, through this study we provide experimental evidence that existing implants are very diverse but also seriously overdesigned embedded systems. They address medical applications through ad-hoc device implementations which are lacking a systematic design approach. A more structured and top-down approach needs to be asserted if we want to exploit the benefits microelectronics technology has to offer these days.

One step towards this direction is the careful design of a generic processor family $\mathbb{P}$ which can service a wide number of applications. This generic processor

family must have at least one processor from $\{10, 13, 14, 15, 17, 18, 20\}$ in order to satisfy applications #A, #E and #B; and configuration $\{2\}$ in order to satisfy #C, #D and #F. Out of the former set, we observe that $\{15\}$ has the least area and $\{20\}$ the least power. Since area and power are both of primary concern in a constrained implant, the generic-processor family may contain both these processors. The implant designer may, then, choose either of these processors depending on which of these two constraints is more pressing for the (unknown) application in question. Therefore, the family of processors chosen is $\{2, 15, 20\}$.

6.3.6 Conclusions

In this work, we have presented a complete approach towards systematic, educated and automated microarchitectural specification of processors for biomedical, microelectronic implants. We have provided 19 Pareto-optimal processor alternatives investigating a large set of hardware parameters such as I-cache and D-cache geometries, branch-prediction policy and memory latency. To the best of our knowledge, we have also provided the first comparison between the suggested processor configurations and existing, documented implantable devices across a wide range of applications. To manage this, we have established means of direct comparison based on careful assumptions that take into account the unavoidable inaccuracies of our tools. In doing so, we have proposed processors that can operate under worst-case conditions, i.e. they are suitably provisioned for the mission-critical implant applications.

6.4 Summary

In many ways, this chapter has been the epitome of the current thesis' efforts in proposing a new processor for biomedical, microelectronic implants. First off, through development of the ImpEDE framework, we have offered to the community a freely available, new tool for performing DSE of implant processors – and any other processors fitting our application field, for that matter. The tool has been designed efficiently and a second, parallelized version allows for significantly reduced simulation times.

Hands-on use of ImpEDE has, in turn, made us realize omissions in the previously established ImpBench suite and have extended the suite with modified versions of the existing benchmarks. ImpEDE has, yet once more, been put to good use in the first – to our knowledge – characterization of (ImpBench) com-

ponents based on the different directions they push the GA-based processor optimization process.

Last, a crucial goal of this thesis work has largely been achieved: We have employed ImpEDE to drive a full-blown design-space exploration for pinpointing optimal SiMS-processor microarchitectural configurations. Nineteen evolved solutions have been juxtaposed with six diverse, actual implantable devices drawn from our previous implant survey. With performance, power, energy and area constraints in mind, three of those optimal (SiMS) solutions have been shown capable of seamlessly replacing the processing/controlling components of all six considered implantable devices.

Note. The content of this chapter is based on collaborative work with D. Dave, ir., and has resulted in the following papers:

D. Dave, C. Strydis, G. N. Gaydadjiev, **ImpEDE: A Multidimensional Design-Space Exploration Framework for Biomedical-Implant Processors**, 21st IEEE International Conference on Application-specific Systems Architectures and Processors (ASAP'10), pp. 39-46, Rennes, France, July 2010.

C. Strydis, D. Dave, G. N. Gaydadjiev, **ImpBench Revisited: An Extended Characterization of Implant-Processor Benchmarks**, International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS X), pp. 126-135, Samos, Greece, July 2010.

C. Strydis, D. Dave, **Identifying Optimal Generic Processors for Biomedical Implants**, 28th IEEE International Conference on Computer Design (ICCD'10), pp. 494-501, Amsterdam, The Netherlands, October 2010.

7 Conclusions

THIS dissertation has served as the first attempt towards a new design paradigm for biomedical implants and, in particular, the digital processors thereof. In the process of specifying this paradigm, we have become occupied with a range of diverse topics. Starting from an in-depth survey, taxonomy and analysis of implantable devices, we have identified underlying trends in the studied field and have used our findings (along with our intuitions) to establish the SiMS project which effectively is the research platform for this new implant-design paradigm. Of the whole platform, we have singled out and occupied ourselves with the SiMS processor, an envisioned universal, low-power and dependable processor for implant applications, and a cornerstone piece in the SiMS ensemble at that. In the absence of any reference designs or design tools in this brave new world of implants, we have spent considerable time detailing the specifics of a suitable exploration framework for the SiMS processor. In the process, we have produced interesting scientific results and have, in parallel, developed methods and tools which have been iteratively refined and are, in themselves, significant contributions in the field. Finally, we have performed a first, complete design-space-exploration study and have procured a number of optimal SiMS-processor microarchitectures, able to seamlessly serve an array of actual, reported implant applications.

The current chapter concludes this dissertation by giving a concise summary of the research work performed. It, then, enumerates the major contributions delivered and ends with a discussion of the still open issues of this work and the author's suggestions regarding future research directions in the field of microelectronic implants from the viewpoint of SiMS.

7.1 Outlook

Chapter 2 has presented a taxonomy and in-depth analysis of a large number of implantable systems covering the 20-year-long period 1994–2005. Numerous device parameters have been investigated, resulting in a detailed classification. Interesting and, at times, counter-intuitive results have been drawn. One such result is that, while modern microelectronic implants can be effectively grouped in only two main categories in terms of functionality, with similar power and other requirements, yet, most existing implants constitute ad-hoc designs with minimal design reuse. Another surprising finding is the net increase in the dynamic power consumption of implants. A third one is the drop in reliability provisions over the years. Behind such effects we believe that lies the fact that implant designs are slowly moving from FSM-based to software-based systems. These and other lessons learned throughout the course of this survey have convinced us of the imperative need to introduce a more structured and conscious design paradigm for implants.

In Chapter 3 we have taken up the task of introducing for the first time the novel SiMS concept. We have outlined the various aspects of the SiMS framework and, then, moved to presenting the organization of our work on the SiMS processor, the exploration of which is the main focus in this dissertation. For this exploration, certain knowledge and tools were needed to be in place. Accordingly, we have concluded the chapter by establishing all needed background knowledge (combined with the knowledge acquired through the survey in Chapter 2). An extensive list of related works on all topics pertaining to the SiMS-processor exploration has been presented.

Chapter 4 has put all background knowledge to good use: A cycle-accurate, power- and performance-simulator (XTREM) has been selected and its suitability for the purpose intended has been demonstrated. Through use of this simulator, a large-scale investigation of suitable workloads for future implant processors has been conducted. This investigation, apart from highlighting best-suited benchmarks from various areas (compression, encryption etc.) and across various metrics (performance, power, energy, memory footprint etc.) has also offered ample hints towards the optimal design of the SiMS processor. Furthermore, it has led to the creation of ImpBench, a novel benchmark suite that aspires to be a reference platform for designing and comparing implant processors. Last but not least, in this chapter a case study of a first, complete application to be run on the envisioned SiMS processor has been detailed.

In Chapter 5, with all needed tools (simulator, benchmarks, datasets) finally in

place, we have explored two particular microarchitectural aspects of the envisioned SiMS processor, namely, the cache and branch-prediction subsystems. The results have revealed that – contrary to general belief – incorporation of such advanced structures in a minimalistic processor, such as the SiMS processor, sometimes does introduce benefits. Apart from the novel application field, what makes this investigation more important is the fact that we studied the effects of various configurations of the two subsystems (and their interplay) with respect to the whole processor core. Total performance, power, energy and area metrics have been utilized to get the complete picture when considering such subsystems in an implant processor (or any processor, for that matter). Last but not least, the needs of the aforementioned exploration processes have led us to extend the XTREM-based simulation environment with CACTI and other automation and support scripts. The new software ensemble allows for automatic configuring, simulating, acquiring and aggregating the generated data for the purposes of any microarchitectural exploration.

Last, Chapter 6 has taken this software ensemble a step further. By incorporating a Genetic Algorithm for automatically traversing the design space and by better tuning of the components, we have developed the ImpEDE framework, a freely available, new tool for performing DSE of implant processors – and any other processors fitting our application field. The tool has been designed efficiently and a second, parallelized version allows for significantly reduced simulation times. Through use of ImpEDE we have taken a second step forward by extending the ImpBench suite (v1.1) and by offering a novel means of characterizing benchmark kernels, through the pattern they exhibit when used to evolve optimal processor solutions. As the last step of this chapter, and a culminating point in this dissertation, we have employed ImpEDE to drive a full-blown design-space exploration for pinpointing optimal SiMS-processor microarchitectural configurations. Nineteen evolved solutions have been juxtaposed with six diverse, actual implantable devices drawn from our previous implant survey. With performance, power, energy and area constraints in mind, three of those optimal (SiMS) solutions have been shown capable of seamlessly replacing the processing/controlling components of all six considered implantable devices. The first attempt to specify the SiMS-processor microarchitecture has born tangible results.

7.2 Contributions

The main contributions of this dissertation can be summarized as follows:

1. **Holistic approach in analysis and taxonomy of implants and identification of design trends**: This is the first time that an associative analysis of this magnitude takes place. The analysis in Chapter 2 is original in that it suggests a holistic consideration of microelectronic implants and is successful in that it reveals previously undetected, crucial trends.

2. **Holistic approach in synthesis; definition of a new, top-down design paradigm for implantable systems (SiMS)**: This dissertation advocates a holistic approach in the conception and design of modern implants. Operating under the assumption that implants will constitute an important means towards improved, personal healthcare and, in view of observed transitions, we have proposed the SiMS framework which aspires to reduce implant-design risks, costs and time, and to make implant-based treatment more accessible to the general public.

3. **Evaluation of compression and encryption algorithms in the context of implantable systems**: An all-out evaluation of these algorithms across a significant number of metrics has led to two benefits; first, this study can be used to choose suitable algorithms for various fields of embedded systems apart from implants (e.g. WSNs, mobiles). Second, this study has offered insights on frequently seen instructions in the processor. In effect, a better understanding of the architectural needs of modern embedded systems can be gained, as was the case for implants.

4. **Establishment of a new benchmark suite (ImpBench) and a derived worst-case benchmark mix for evaluating different implant-processor designs**: The collection of benchmarks in ImpBench is the first attempt to define a reference point in the unstructured field of implant design. Future repeatable processor designs can quickly be evaluated against ImpBench (and the worst-case benchmark mix if real-time deadlines are known) and can be modified early on in the design phase, with minimal penalty.

5. **Quantification of complex effects between the L1 I/D-caches and the BRPED policy, from the processor point of view**: The interplay among the caches, the BPRED policy and the other components of the processor exhibit non-linear phenomena. Under certain conditions, it

has been shown that relatively expensive cache and/or BPRED configurations can lead to globally better processors for implant applications.

6. **Deployment of a new simulation and exploration frameworks (ImpEDE) primarily for exploring implant-processor characteristics and, secondarily, for further characterization of benchmark collections**: The framework provides the processor designer with evolved Pareto fronts through which informed decisions can be made about specific implant families after analyzing their particular tradeoffs and requirements. A highly efficient, parallelized version of ImpEDE has also been created to evolve the fronts and has as its objectives the optimization of power, performance and area. In addition, the extensibility of our framework has been illustrated by modifying it to include a case study of a synthetic implant application with hard realtime deadlines. ImpEDE can be used as efficiently in any other niche of embedded systems.

7. **First, automatic, multiobjective design-space exploration of optimal SiMS processor microarchitectures**: This DSE has been a successful proof-of-concept optimization attempt for the SiMS processor. We have chosen processor configurations from the Pareto-optimal processor set found by the DSE using real implants as case studies. We have found that, even under the extremely biased constraints that we use, our SiMS processors perform better than many of the real implants. This provides strong hints towards designing an implant processor that is generic enough to cover most, if not all, implant applications.

7.3 Open Issues and Future Directions

When compared to other fields involving computer engineering such as mobile telephony and portable computing, the field of biomedical implants is still in its infancy. This fact has been the major motivation and, at the same time, the major setback in this dissertation work. To deal with the severe lack of prior art in the field, at times we had to take some decisions based on experience or intuition and to make some modeling assumptions, all detailed in the previous chapters. Such decisions have also been dictated by prosaic reasons such as the limited capabilities provided by our simulation tools. We also had to somewhat restrict our research-work goals in order to efficiently deal with a few, specific problems and a field which is still largely uncharted.

For all the above reasons and given the limited resources and time frame at our

disposal, a number of open issues in this dissertation work remain. In what follows, we discuss these issues and offer future directions for research in the field.

A first, future direction of this work is the **extension of the implant survey** with more recent systems. Since 2005 (that the survey ends), a number of new devices has probably been released. It would be interesting to see how these newer systems look like and whether they fall on our predictions based on the previous years.

ImpBench is a dynamic construct with a wide range of potential implant applications. Even so, in its present state it is far from complete. In the future, more benchmarks should be added, subject to ongoing research in the field and access to more resources (e.g. executed implant code). Among others, we anticipate simple DSP applications as potential candidates. We believe that more implant application- or domain-specific algorithms should be evaluated (except for compression and encryption, that is), yet one should be careful not to become too specific for fear of losing this much desired universal processor approach. Also, more "real applications" like the ones (DMU, motion) we already included should be added. Nevertheless, these are simply program kernels that cannot interact with implant peripherals. So far, we have emulated interaction with the outside world through the use of file I/O (e.g. DMU). It would be interesting to establish benchmarks (and assorted simulation environments) that do, somehow, interact with real-world interfaces.

Although **reliability** is one of the major reasons for the need to design processors specifically for implants, this dissertation does not directly address reliability, due mainly to time limitations. In the future, we intend to expand our DSE framework to also optimize for system reliability in order to ensure error-free operation of critical implant applications. For this, we need to introduce a fourth metric based on reliability, and expand our tools accordingly. Work has already begun on porting XEEMU to our system as a more bug-free and accurate replacement for XTREM. We would also like to expand the simulator models with more parameters such as (off-chip) memory, effectively allowing for SoC explorations. Finally, we would also like to include more real-life applications in our studies – however, this is influenced by the extremely limited information released in this field.

The careful observer may have noticed that, although suggestions are made on **ISA optimizations** for the SiMS processor in Chapter 4, nonetheless, our subsequent optimizations are limited to microarchitectural optimizations only. The reason for this is that modification of the processor ISA along

with the assorted binary utilities and compiler was virtually impossible for the XTREM simulator. Of late, we have become licensed users of the rapid architecture-exploration tools called Processor and Compiler Designer, by Synopsys. Through these tools, experimentation with multiple ISA variants on a given processor becomes fast. Unfortunately, this tool was not available at the onset of our SiMS-processor experiments and only lately has it been put to good use.

Since the SiMS framework is – at least, at the moment of writing – envisioned as a SoC, it becomes apparent that a sort of **"implant interconnect"** is also required. Outright properties of such an interconnect are expected to be low power consumption, small area footprint, low pin count and, yet, communication robustness. Although some work could be borrowed from the field of WSNs, nevertheless, there is currently no interconnect explicitly designed to address implant SoCs. No interconnect scheme from any other field (e.g. WSNs, automotives) can be adopted in the implant domain due to the latter's stringent mix of design requirements.

The long-term goal of the SiMS project is silicon, multi-sensor/-actuator, single-chip, wireless medical systems. Such systems will be produced using fully integrated CMOS processes. In addition, they will be capable of **context-sensitive behavior** (thus, smart), due to their multi-parameter awareness and communication abilities. The combination of the aforementioned issues with the envisioned **modular system** approach, introduces even more research challenges.

Bibliography

[1] "EEMBC," www.eembc.com.

[2] 3-WAY, BLOWFISH, DES, GOST, IDEA, RC5 source code, www.cis.udel.edu/~mills/database/schneier/.

[3] B. Abel-Smith and E. Mossialos, "Cost containment and health care reform: a study of the european union," *Health Policy*, vol. 28, no. 2, pp. 89–132, 1994.

[4] T. Akin and K. Najafi, "A wireless implantable multichannel digital recording system for a micromachined sieve electrode," in *IEEE Journal of Solid-State Circuits*, vol. 33, Jan. 1998, pp. 109–118.

[5] K. Arabi and M. Sawan, "A monolithic miniaturized programmable implant for neuromuscular stimulation," in *IEEE 17th Annual Conference Engineering in Medicine and Biology Society (EMBS)*, Montreal, Quebec, Canada, 20-23 September 1995, pp. 1131–1132.

[6] G. Ascia, V. Catania, and M. Palesi, "Parameterised system design based on genetic algorithms," in *CODES '01: Proceedings of the ninth international symposium on Hardware/software codesign*. New York, NY, USA: ACM, 2001, pp. 177–182.

[7] K. Au-Yeung, C. Johnson, and P. Wolf, "A novel implantable cardiac telemetry system for studying atrial fibrillation," in *Physiological Measurement*, 11 August 2004, pp. 1223–1238.

[8] T. Austin, E. Larson, and D. Ernst, "Simplescalar: an infrastructure for computer system modeling," *IEEE Computer*, vol. 35, no. 2, pp. 59–67, February 2002.

[9] T. Back, "Optimal mutation rates in genetic search," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, 1993, pp. 2–8.

[10] T. Bapty, S. Neema, J. Scott, J. Sztipanovits, and S. Asaad, "Model-integrated tools for the design of dynamically reconfigurable systems," ISIS, Vanderbilt University, Tech. Rep., 2000.

[11] K. Barr and K. Asanovic, "Energy-aware lossless data compression," *ACM Transactions on Computer Systems*, vol. 24, no. 3, pp. 250–291, August 2006.

[12] S. Baskiyar, "A real-time fault tolerant intra-body network," in *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN'02)*, 6-8 November 2002, pp. 235 – 240.

[13] R. Beach, F. Kuster, and F. Moussy, "Subminiature implantable potentiostat and modified commercial telemetry device for remote glucose monitoring," in *IEEE Transactions on Instrumentation and Measurement*, vol. 48, Dec. 1999, pp. 1239–1245.

[14] E. G. Bekele and J. W. Nicklow, "Multi-objective automatic calibration of swat using nsga-ii," *Journal of Hydrology*, vol. 341, no. 3-4, pp. 165 – 176, 2007.

[15] J. Berkman and J. Prak, "Biomedical microprocessor with analog i/o," in *IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, 19 February 1981, pp. 168–169.

[16] R. Braden, D. Borman, and C. Partridge, "Computing the internet checksum," *SIGCOMM Comput. Commun. Rev.*, vol. 19, no. 2, pp. 86–94, 1989.

[17] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," *SIGARCH Comput. Archit. News*, vol. 28, no. 2, pp. 83–94, 2000.

[18] D. Burger and T. Austin, "The simplescalar tool set, version 2.0," *ACM SIGARCH Computer Architecture News*, vol. 25, no. 3, pp. 13–25, June 1997.

[19] Cell Relay Retreat: CRC-32 Calculation, Test Cases and HEC Tutorial, http://cell.onecall.net/cell-relay/publications/software/.

[20] C.-C. Chang, S. Muftic, and D. Nagel, "Measurement of energy costs of security in wireless sensor nodes," in *16th International Conference on Computer Communications and Networks (ICCCN)*, 2007, pp. 95–102.

[21] C. Cho, W. Zhang, and T. Li, "Informed Microarchitecture Design Space Exploration Using Workload Dynamics," in *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2007, pp. 274–285.

[22] J. Christensen and T. Pallesen, "Institutions, distributional concerns, and public sector reform," *European Journal of Political Research*, vol. 39, no. 2, pp. 179–202, 2001.

[23] G. Contreras and M. Martonosi, "The XTREM Power and Performance Simulator for the Intel XScale Core: Design and Experiences," *ACM Transactions on Embedded Computing Systems*, vol. 6, no. 1, pp. 1–25, February 2007.

[24] G. Contreras, M. Martonosi, J. Peng, R. Ju, and G.-Y. Lueh, "XTREM: A Power Simulator for the Intel XScale Core," in *LCTES'04*, 2004, pp. 115–125.

[25] P. Cross, R. Kunnemeyer, C. Bunt, D. Carnegie, and M. Rathbone, "Control, communication and monitoring of intravaginal drug delivery in dairy cows," in *International Journal of Pharmaceuticals*, vol. 282, 10 September 2004, pp. 35–44.

[26] T. Custers, O. A. Arah, and N. S. Klazinga, "Is there a business case for quality in the netherlands? a critical analysis of the recent reforms of the health care system," *Health Policy*, vol. 82, no. 2, pp. 226–239, 2007.

[27] D. Dave, C. Strydis, and G. N. Gaydadjiev, "Impede: A multidimensional design-space exploration framework for biomedical-implant processors," in *To appear in: Proceedings of the 21th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP'10)*, July 7-9 2010.

[28] D. Dave, "Automated implant-processor design: An evolutionary multiobjective exploration framework," Master's thesis, TU Delft, 2010.

[29] N. de Vries, "Lossless data-compression kit, lds v1.3," www.nicodevries.com/nico/lds13.zip.

[30] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, LTD, 2001.

[31] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast elitist multi-objective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, 2000.

[32] D. DLima, C. Townsend, S. Arms, B. Morris, and C. C. Jr, "An implantable telemetry device to measure intra-articular tibial forces," in *Journal of Biomechanics*, vol. 38, Feb. 2005, pp. 299–304.

[33] W. Du, J. Deng, Y. Han, P. Varshney, J. Katz, and A. Khalili, "A pairwise key predistribution scheme for wireless sensor networks," *ACM Transactions on Information and System Security (TISSEC)*, vol. 8, pp. 228–258, May 2005.

[34] L. Eccles, "A brief description of ieee p1451.2," in *Proceedings of the Sensors Expo*, Oct. 1997, pp. 81–90.

[35] H. Ector and P. Vardas, "Heart disease and stroke statistics - 2008 update (at-a-glance version)," *American Heart Association*, 2008.

[36] T. Eggers, C. Marschner, U. Marschner, B. Clasbrummel, R. Laur, and J. Binder, "Advanced hybrid integrated low-power telemetric pressure monitoring system for biomedical applications," in *IEEE Proceedings of Microelectromechanical Systems (MEMS'00)*, Miyuzaki, Japan, 2000, pp. 329–334.

[37] ——, "Wireless intra-ocular pressure monitoring system integrated into an artificial lens," in *Proceedings of the 1st Annual International IEEE-EMBS Special Topic Conference on Microtechnologies in Medicine & Biology*, Lyon, France, 12-14 October 2000, pp. 466–469.

[38] C. Enokawa, Y. Yonezawa, H.Maki, and M. Aritomo, "A microcontroller-based implantable telemetry system for sympathetic nerve activity and ecg measurement," in *Proceedings of the 19th Annual International Conference of the IEEE in Engineering in Medicine and Biology Society (EMBS)*, vol. 5, Chicago, Illinois, USA, 30 October - 2 November 1997, pp. 2232–2234.

[39] M. Ericson, M. Wilson, G. Cote, C. Britton, W. Xu, J. Baba, M. Bobrek, M. Hileman, M. Moore, and S. Frank, "Development of an implantable oximetry-based organ perfusion sensor," in *Proceedings of the 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, vol. 2, San Francisco, California, USA, 1-5 September 2004, pp. 2235–2238.

[40] S. Eyerman, L. Eeckhout, and K. De Bosschere, "Efficient design space exploration of high performance embedded out-of-order processors," in *DATE '06: Proceedings of the conference on Design, automation and test in Europe*. 3001 Leuven, Belgium, Belgium: European Design and Automation Association, 2006, pp. 351–356.

[41] K. Fernald, T. Cook, T. M. III, and J. Paulos, "A microprocessor-based implantable telemetry system," in *IEEE Computer*, vol. 24, Mar. 1991, pp. 23–30.

[42] K. Fernald, B. Stackhouse, J. Paulos, and T. Miller, "A system architecture for intelligent implantable biotelemetry instruments," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, vol. 11, Nov. 1989, pp. 1411–1412.

[43] L. Ferrigno, S. Marano, V. Paciello, and A. Pietrosanto, "Balancing computational and transmission power consumption in wireless image sensor networks," in *VECIMS'05*, July 2005, pp. 61–66.

[44] A. Flammini, P. Ferrari, E. Sisinni, D. Marioli, and A. Taroni, "Sensor interfaces: from field-bus to ethernet and internet," in *Sensors and Actuators A: Physical*, vol. 101, 2002, pp. 194–202.

[45] B. Flick and R. Orglmeister, "A portable microsystem-based telemetric pressure and temperature measurement unit," in *IEEE Transactions on Biomedical Engineering*, vol. 47, Jan. 2000, pp. 12–16.

[46] W. Fornaciari, D. Sciuto, C. Silvano, and V. Zaccaria, "A design framework to efficiently explore energy-delay tradeoffs," in *CODES '01: Proceedings of the ninth international symposium on Hardware/software codesign*, New York, NY, USA, 2001, pp. 260–265.

[47] G. Gaubatz, J.-P. Kaps, and B. Sunar, "Public key cryptography in sensor networksrevisited," in *Lecture Notes in Computer Science*, 2005, pp. 2–18.

[48] M. Geelnard, "Basic compression library, bcl v1.2.0," http://bcl.comli.eu/.

[49] M. Geisler and D. Jeutter, "An implantable transcutaneously programmable and rechargeable nerve regenerator with telemetry links," in *Proceedings of the 15st Annual International Conference of the IEEE on Engineering in Medicine and Biology Society (EMBS)*, 28-31 October 1993, pp. 1240–1241.

[50] M. Ghovanloo and K. Najafi, "A modular 32-site wireless neural stimulation microsystem," in *IEEE Journal of Solid-State Circuits*, vol. 39, December 2004, pp. 2457–2466.

[51] F. Gielen, "Deep brain stimulation: Current practice and challenges for the future," in *Proceedings of the 1st International IEEE EMBS Conference on Neural Engineering*, Capri Island, Italy, 20-22 March 2003, pp. 489–491.

[52] T. Givargis, F. Vahid, and J. Henkel, "Evaluating power consumption of parameterized cache and bus architectures in system-on-a-chip designs," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 9, no. 4, pp. 500–508, Aug 2001.

[53] M. Gries, "Methods for evaluating and covering the design space during early design development," *Integr. VLSI J.*, vol. 38, no. 2, pp. 131–183, 2004.

[54] J. Grossschadl, S. Tillich, C. Rechberger, M. Hofmann, and M. Medwed, "Energy evaluation of software implementations of block ciphers under memory constraints," in *Conference on Design, automation and test in Europe*, 2007, pp. 1110–1115.

[55] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown, "Mibench: A free, commercially representative embedded benchmark suite," *IEEE International Workshop on Workload Characterization*, pp. 3–14, 2 December 2001.

[56] C. Harrigal and R. Walters, "The development of a microprocessor controlled implantable device," in *IEEE Proceedings of the 1990 Sixteenth Annual Northeast Bioengineering Conference*, Mar. 1990, pp. 137–138.

[57] C. Harrigal, R. Walters, and R. Reynolds, "An implanted device for stimulating paralyzed vocal chords," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, 29 October - 1 November 1992, pp. 1368–1369.

[58] G. Hekstra, G. La Hei, P. Bingley, and F. Sijstermans, "TriMedia CPU64 design space exploration," in *iccd*. Published by the IEEE Computer Society, 1999, p. 599.

[59] J. Hennessy and D. Patterson, *Computer Architecture - A Quantitative Approach*, 4th ed., D. Penrose, Ed. Morgan Kaufmann, 2003.

[60] Z. Herczeg, A. Kiss, D. Schmidt, N. Wehn, and T. Gyimóthy, "XEEMU: An Improved XScale Power Simulator," *Integrated Circuit and System Design - Power and Timing Modeling, Optimization and Simulation (PATMOS'07)*, pp. 300–309, 2007.

[61] P. Hicks, M. Walnock, and R. Owens, "Analysis of power consumption in memory hierarchies," *Low Power Electronics and Design, 1997. Proceedings., 1997 International Symposium on*, pp. 239–242, Aug 1997.

[62] K. Hille, J. Draeger, T. Eggers, and P. Stegmaier, "Technical construction, calibration and results with a new intraocular pressure sensor with telemetric transmission [article in german]," in *Klinische Monatsblatter fur Augenheilkunde*, vol. 218, May 2001, pp. 376–380.

[63] J. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–72, 1992.

[64] *Intel XScale Core Developer's Manual*, Intel, December 2000.

[65] *Intel XScale Microarchitecture for the PXA255 Processor: User's Manual*, Intel Corporation, March 2003.

[66] International Technology Roadmap for Semiconductors (ITRS), "Available: www.itrs.net/common/2004update/2004update.htm," 2004.

[67] T. Jansen and I. Wegener, "On the choice of the mutation probability for the (1+1) ea," *Lecture notes in computer science*, pp. 89–98, 2000.

[68] Y.-S. Jeong and S.-H. Lee, "Hybrid key establishment protocol based on ecc for wireless sensor network," *Lecture Notes in Computer Science*, vol. 4611, pp. 1233–1242, August 2007.

[69] D. Jimenez, S. Keckler, and C. Lin, "The impact of delay on the design of branch predictors," *Microarchitecture, 2000. MICRO-33. Proceedings. 33rd Annual IEEE/ACM International Symposium on*, pp. 67–76, 2000.

[70] D. Jones, "Application of splay trees to data compression," *Communications of the ACM*, vol. 31, no. 8, pp. 996–1007, August 1988.

[71] M. Kamble and K. Ghose, "Analytical energy dissipation models for low power caches," *Low Power Electronics and Design, 1997. Proceedings., 1997 International Symposium on*, pp. 143–148, Aug 1997.

[72] V. Kathail, S. Aditya, R. Schreiber, B. R. Rau, D. C. Cronquist, and M. Sivaraman, "Pico: Automatically designing custom computers," *Computer*, vol. 35, pp. 39–47, 2002.

[73] S. Kawahito, S. Ueda, M. Ishida, T. Nakamura, S. Usui, and S. Nagaoka, "A cmos integrated circuit for multichannel multiple-subject biotelemetry using bidirectional optical transmissions," in *IEEE Transactions on Biomedical Engineering*, vol. 41, Apr. 1994, pp. 400–406.

[74] W. Kickert, "Public governance in the netherlands: An alternative to anglo-american managerialism," *Public administration*, vol. 75, no. 4, pp. 731–752, 1997.

[75] I. Kim and O. de Weck, "Adaptive weighted sum method for multiobjective optimization: a new method for Pareto front generation," *Structural and Multidisciplinary Optimization*, vol. 31, no. 2, pp. 105–116, 2006.

[76] N. Kimura and S. Latifi, "A survey on data compression in wireless sensor networks," in *ITCC'05*, 2005, pp. 8–13.

[77] H. Lanmüller, S. Sauermann, E. Unger, G. Schnetz, W. Mayr, M. Bijak, D. Rafolt, and W. Girsch, "Battery-powered implantable nerve stimulator for chronic activation of two skeletal muscles using multichannel techniques," in *Artificial Organs*, vol. 23, May 1999, pp. 399–402.

[78] H. Lanmüller, E. Unger, M. Reichel, Z. Ashley, W. Mayr, and A. Tschakert, "Implantable stimulator for the conditioning of denervated muscles in rabbit," in *Proceedings of the 8th Vienna International Workshop on Functional Electrical Stimulation*, Vienna, Austria, 10-13 September 2004.

[79] Y. Law, J. Dourmen, and P. Hartel, "Survey and benchmark of block ciphers for wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 2, pp. 65–93, February 2006.

[80] C. Lee, M. Potkonjak, and W. Mangione-Smith, "Mediabench: a tool for evaluating and synthesizing multimedia and communications systems," *30th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 330–335, 1-3 Dec 1997.

[81] K. Lee, "Ieee 1451:a standard in support of smart transducer networking," in *Proceedings of the 17th IEEE Instrumentation and Measurement Technology Conference (IMTC 2000)*, vol. 2, Jan. 2000, pp. 525–528.

[82] A. Lenstra and E. Verheul, "Selecting cryptographic key sizes," *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, vol. 14, no. 4, pp. 255–293, 2001.

[83] R. Lerch, E. Spiegel, R. H. R. Kakerow, H. Kappert, H. Kohlhaas, N. Kordas, M. Buchmann, T. Franke, Y. Manoli, and J. Muller, "A programmable mixed-signal asic for data acquisition systems in medical implants," in *IEEE International Solid-State Circuits Conference*, vol. 38, Piscataway, New Jersey, USA, 1995, pp. 162–163.

[84] C. Liang, J. Chen, C. Chung, C. Cheng, and C. Wang, "An implantable bi-directional wireless transmission system for transcutaneous biological signal recording," *Physiological Measurement*, vol. 26, pp. 83–97, February 2005.

[85] X. Luo, K. Zheng, Y. Pan, and Z. Wu, "Encryption algorithms comparisons for wireless networked sensors," in *IEEE International Conference on Systems, Man and Cybernetics*, 2004, pp. 1142–1146.

[86] D. Maniezzo, K. Yao, and G. Mazzini, "Energetic trade-off between computing and communication ressource in multimedia surveillance sensor network," in *4th Int' Workshop on Mobile and Wireless Communications Network*, 2002, pp. 373–376.

[87] A. Mason, N. Yazdi, N. Najafi, and K. Wise, "A low-power wireless microinstrumentation system for environmental monitoring," in *The 8th International Conference on Solid-State Sensors and Actuators, and Eurosensors IX (Transducers'95)*, vol. 1, 25-29 June 1995, pp. 107–110.

[88] G. Mavrotas, "Effective implementation of the [epsilon]-constraint method in multi-objective mathematical programming problems," *Applied Mathematics and Computation*, vol. 213, no. 2, pp. 455 – 465, 2009.

[89] Z. McCreesh and N. Evans, "Radio telemetry of vaginal temperature," in *Proceedings of the 16th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, vol. 2, 1994, pp. 904–905.

[90] Z. McCreesh, N. Evans, and W. Scanlon, "Vaginal temperature sensing using uhf radio telemetry," in *Medical Engineering & Physics*, vol. 18, 1996, pp. 110–114.

[91] Medtronic - Cardiology product list, www.medtronic.com/physician/cardiology.html.

[92] G. Memik, W. H. Mangione-Smith, and W. Hu, "Netbench: a benchmarking suite for network processors," in *IEEE/ACM international conference on Computer-aided design (ICCAD'01)*, Piscataway, NJ, USA, 2001, pp. 39–42.

[93] T. Miller, B. Bhuva, R. Barnes, J. Duh, H. Lin, and D. V. den Bout, "The hector microprocessor," in *Proceedings of the IEEE International Conference on Computer Design (ICCD)*, 1986, pp. 406–411.

[94] M. Min, T. Parve, V. Kukk, and A. Kuhlberg, "An implantable analyzer of bio-impedance dynamics - mixed signal approach," in *IEEE Instrumentation and Measurement*, Budapest, Hungary, 21-23 May 2001, pp. 38–43.

[95] S. Mohanty, V. K. Prasanna, S. Neema, and J. Davis, "Rapid design space exploration of heterogeneous embedded systems using symbolic search and multi-granular simulation," in *LCTES/SCOPES '02: Proceedings of the joint conference on Languages, compilers and tools for embedded systems*. New York, NY, USA: ACM, 2002, pp. 18–27.

[96] P. Mohseni and K. Najafi, "Wireless multichannel biopotential recording using an integrated fm telemetry circuit," in *Proceedings of the 26th Annual International Conference of the IEEE in Engineering in Medicine and Biology Society (EMBS)*, San Francisco, CA, USA, 1-5 September 2004, pp. 4083–4086.

[97] J. Mouine, K. Ammar, and Z. Chtourou, "A completely programmable and very flexible implantable pain controller," in *Proceedings of the 22nd Annual EMBS International Conference*, Chicago IL., USA, Jul. 2000, pp. 603–622.

[98] J. Moyne, N. Najafil, D. Judd, and A. Stock, "Analysis of sensor/actuator bus interoperability standard alternatives for semiconductor manufacturing," in *Proceedings of the Sensors Expo Conference*, May 1995.

[99] N. Najafi and K. Wise, "An organization and interface for sensor-driven semiconductorprocess control systems," in *IEEE Transactions on Semiconductor Manufacturing*, vol. 3, Nov. 1990, pp. 230–238.

[100] M. Nelson, "Ddj data compression contest results," *Dr. Dobb's Journal*, vol. 16, no. 11, pp. 62–64, November 1991.

[101] M. Nelson and J.-L. Gailly, *The Data Compression Book, 2nd Edition*. San Mateo, CA: M&T Brooks, 1995.

[102] H. Nys, L. Stultiens, P. Borry, T. Goffin, and K. Dierickx, "Patient rights in EU Member States after the ratification of the Convention on Human Rights and Biomedicine," *Health Policy*, vol. 83, no. 2-3, pp. 223–235, October 2007.

[103] M. Oberhumer, "LZO v2.0.2," www.oberhumer.com/opensource/lzo/.

[104] H. Ohta and M. Matsui, *A Description of the MISTY1 Encryption Algorithm*, United States, 2000.

[105] D. Parikh, K. Skadron, Y. Zhang, and M. Stan, "Power-aware branch prediction: Characterization and design," *IEEE Transactions on Computers*, vol. 53, no. 2, pp. 168–186, 2004.

[106] H. Park, H. Nam, B. Song, and J. Cho, "Design of miniaturized telemetry module for bidirectional wireless endoscopy," *IEICE Transactions on Fundamentals on Electronics, Communications and Computer Sciences*, vol. 6, pp. 1487–1491, June 2003.

[107] J. Park, S. Choi, H. Seo, and T. Nakamura, "Fabrication of cmos ic for telemetering biological signals from multiple subjects," in *Sensors and Actuators, A: Physical*, vol. 43, 1994, pp. 289–295.

[108] S. Pasricha and A. Veidenbaum, "Improving branch prediction accuracy in embedded processors in the presence of context switches," *Computer Design, 2003. Proceedings. 21st International Conference on*, pp. 526–531, Oct. 2003.

[109] J. Pauley, M. Reite, and S. Walker, "An implantable multi-channel biotelemetry system," in *Electroencephalography and Clinical Neurophysiology*, vol. 37, Jan. 1974, pp. 153–160.

[110] A. Pimentel, L. Hertzberger, P. Lieverse, P. van der Wolf, and F. Deprettere, "Exploring embedded-systems architectures with artemis," *Computer*, pp. 57–63, 2001.

[111] S. Pourmehdi, P. Strojnik, P. Peckham, J. Buckett, and B. Smith, "A custom-designed chip to control an implantable stimulator and telemetry system for control of paralyzed muscles," in *Proceedings of the 6th Vienna International Workshop on Functional Electrical Stimulation*, Vienna, Austria, 2224 September 1998.

[112] ——, "A custom-designed chip to control an implantable stimulator and telemetry system for control of paralyzed muscles," in *Artificial Organs*, vol. 23, May 1999, pp. 396–398.

[113] F. Pramassing, D. Puttjer, R. Buss, and D. Jager, "Intraocular vision aid (ios): Optical signal transmission and image generation," in *World Congress on Medical Physics and Biomedical Engineering*, Chicago, USA, 2000.

[114] A. R. Price, I. I. Voutchkov, G. E. Pound, N. R. Edwards, T. M. Lenton, and S. J. Cox, "Multiobjective tuning of grid-enabled earth system models using a non-dominated sorting genetic algorithm (nsga-ii)," in *E-SCIENCE '06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*. Washington, DC, USA: IEEE Computer Society, 2006, p. 117.

[115] D. Rollins, C. Killingsworth, G. Walcott, R. Justice, R. Ideker, and W. Smith, "A telemetry system for the study of spontaneous cardiac arrhythmias," in *IEEE Transactions on Biomedical Engineering*, vol. 47, Jul. 2000, pp. 887–892.

[116] C. Sadler and M. Martonosi, "Data compression algorithms for energy-constrained devices in delay tolerant networks," in *SenSys'06*, November 2006, pp. 265–278.

[117] S. Salmons, G. Gunning, I. Taylor, S. Grainger, D. Hitchings, J. Blackhurst, and J. Jarvis, "Asic or pic ? implantable stimulators based on semi-custom cmos technology or low-power microcontroller architecture," in *Medical Engineering & Physics*, vol. 23, 2001, pp. 37–43.

[118] R. Sanders and M. Lee, "Implantable pacemakers," in *Proceedings of the IEEE*, vol. 84, Mar. 1996, pp. 480–486.

[119] M. Sawan, S. Robin, B. Provost, Y. Eid, and K. Arabi, "A wireless implantable electrical stimulator based on two fpgas," in *Proceedings of the IEEE International Conference on Electronic Circuits and Systems (ICECS)*, vol. 2, Piscataway, New Jersey, USA, 1996, pp. 1092–1095.

[120] M. Schwarz, L. Ewe, R. Hauschild, B. Hosticka, J. Huppertz, S. Kolnsberg, W. Mokwa, and H. Trieu, "Single chip cmos imagers and flexible microelectronic stimulators for a retina implant system," in *Sensors and Actuators A: Physical*, vol. 83, 22 May 2000, pp. 40–46.

[121] J. Sears and J. Naber, "Development of a biotelemetric heart valve monitor using a 2.45 ghz transceiver, microcontroller, a/d converter, and sensor gain amplifiers," in *Proceedings of The First Joint BMES/EMBS Conference*, vol. 2, Atlanta, GA, USA, 13-16 October 1999, p. 794.

[122] W. Shiue and C. Chakrabarti, "Memory exploration for low power, embedded systems," in *DAC'99*, 1999, pp. 140–145.

[123] M. Shults, R. Rhodes, S. Updike, B. Gilligan, and W. Reining, "A telemetry-instrumentation system for monitoring multiple subcutaneously implanted glucose sensors," in *IEEE Transactions on Biomedical Engineering*, vol. 41, Oct. 1994, pp. 937–942.

[124] K. Skadron, P. Ahuja, M. Martonosi, and D. Clark, "Branch prediction, instruction-window size, and cache size: performance trade-offs and simulation techniques," *Computers, IEEE Transactions on*, vol. 48, no. 11, pp. 1260–1281, Nov 1999.

[125] SKIPJACK, LOKI91 source code, www.mirrors.wiretapped.net/security/cryptography/algorithms/skipjack/.

[126] S. Slijepcevic, M. Potkonjak, V. Tsiatsis, S. Zimbeck, and M. Srivastava, "On communication security in wireless ad-hoc sensor networks," *Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'02)*, pp. 139–144, 2002.

[127] B. Smith, Z. Tang, M. Johnson, S. Pourmehdi, M. Gazdik, J. Buckett, and P. Peckham, "An externally powered, multichannel, implantable stimulator-telemeter for control of paralyzed muscle," in *IEEE Transactions on Biomedical Engineering*, vol. 45, 1998, pp. 463–475.

[128] SPEC CPU2006, www.spec.org/cpu2006/.

[129] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, pp. 221–248, 1994.

[130] K. Stangel, S. Kolnsberg, D. Hammerschmidt, B. Hosticka, H. Trieu, and W. Mokwa, "A programmable intraocular cmos pressure sensor system implant," in *IEEE Journal of Solid-State Circuits*, vol. 36, Jul. 2001, pp. 1094–1100.

[131] L. Stotts, K. Infinger, J. Babka, and D. Genzer, "An 8 bit microcomputer with analog subsystems for implantable biomedical application," in *IEEE Journal of Solid-State Circuits*, 1989, pp. 292–300.

[132] C. Strydis, G. Gaydadjiev, and S. Vassiliadis, *Implantable microelectronic devices: A comprehensive review*, ser. M.Sc. Thesis. Computer Engineering, Delft University of Technology, July 2005.

[133] ——, "Implantable microelectronic devices: A comprehensive review," Computer Engineering, Delft University of Technology, CE-TR-2006-01, December 2006.

[134] C. Strydis, D. Zhu, and G. Gaydadjiev, "Profiling of symmetric encryption algorithms for a novel biomedical-implant architecture," in *ACM International Conference on Computing Frontiers (CF'08)*, Ischia, Italy, 5-7 May 2008, pp. 231–240.

[135] C. Strydis and G. N. Gaydadjiev, "Profiling of lossless-compression algorithms for a novel biomedical-implant architecture," in *Proceedings of the 6th IEEE/ACM/IFIP international conference on Hardware/Software codesign and system synthesis (CODES+ISSS'08)*, Atlanda, Georgia, USA, 19-24 October 2008, pp. 109–114.

[136] C.-L. Su and A. M. Despain, "Cache design trade-offs for power and performance optimization: a case study," in *ISLPED '95: Proceedings of the 1995 international symposium on Low power design*, New York, NY, USA, 1995, pp. 63–68.

[137] L. Thiele, S. Chakraborty, M. Gries, and S. Künzli, "Design space exploration of network processor architectures," *Network Processor Design: Issues and Practices*, vol. 1, pp. 55–89, 2002.

[138] P. Troyk, I. Brown, W. Moore, and G. Loeb, "Development of bion tm technology for functional electrical stimulation: bidirectional telemetry," in *Proceedings of the 23th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, vol. 4, Istanbul, Turkey, 25-28 October 2001, pp. 1317–1320.

[139] University of Michigan, Sim-Panalyzer 2.0, www.eecs.umich.edu/~panalyzer/.

[140] P. Valdastri, A. Menciassi, A. Arena, C. Caccamo, and P. Dario, "An implantable telemetry platform system for in vivo monitoring of physiological parameters," in *IEEE Transactions on Information Technology in Biomedicine*, vol. 8, September 2004, pp. 271–278.

[141] S. Vassiliadis, J. Phillips, and B. Blaner, "Interlock collapsing alu's," *IEEE Transactions on Computers*, vol. 42, no. 7, pp. 825–839, July 1993.

[142] D. A. V. Veldhuizen and G. B. Lamont, "Evolutionary computation and convergence to a pareto front," in *Genetic Programming 1998: Proceedings of the Third Annual Conference*, J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, Eds. University of Wisconsin, Madison, WI, USA: Morgan Kaufmann, 22-25 Jul. 1998.

[143] R. Venugopalan, P. Ganesan, P. Peddabachagari, A. Dean, F. Mueller, and M. Sichitiu, "Encryption overhead in embedded systems and sensor network nodes: modeling and analysis," in *International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, 2003, pp. 188–197.

[144] A. Wander, N. Gura, H. Eberle, V. Gupta, and S. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," in *3rd IEEE International Conference on Pervasive Computing and Communications*, 2005, pp. 324–328.

[145] L. Wang, P. Hammond, E. Johannessen, T. Tang, A. Astaras, S. Beaumont, A. Murray, J. Cooper, and D. Cumming, "An on-chip programmable instrumentation microsystem for gastrointestinal telemetry applications," in *Proceedings of the 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, San Francisco, California, USA, 1-5 September 2004, pp. 2109–2112.

[146] W. Wang, G. Yan, F. Sun, P. Jiang, W. Zhang, and G. Zhang, "A non-invasive method for gastrointestinal parameter monitoring," in *World Journal of Gastroenterology*, vol. 11, 28 January 2005, pp. 521–524.

[147] J. Warren, R. Dreher, R. Jaworski, J. Putzke, and R. Russie, "Implantable cardioverter defibrillators," in *Proceedings of the IEEE*, vol. 84, 1996, pp. 468–479.

[148] D. Wheeler and R. Needham, "Correction to XTEA," Computer Laboratory, University of Cambridge, Tech. Rep., October 1998.

[149] K. Wise and N. Najafi, "The coming opportunities in microsensor systems," in *Digest of Technical Papers, International Conference on Solid-State Sensors and Actuators (TRANSDUCERS '91)*, San Francisco, California, USA, 24-27 August 1991, pp. 2–7.

[150] T. Wolf and M. Franklin, "Commbench-a telecommunications benchmark for network processors," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS'00)*, Washington, DC, USA, 2000, pp. 154–162.

[151] S. Woods, "The ieee-p1451 transduced to microprocessor interface," in *Sensors Magazine*, June 1996, pp. 43–48.

[152] P. Wouters, M. D. Cooman, D. Lapadatu, and R. Puers, "A low power multi-sensor interface for injectable microprocessor-based animal monitoring system," in *Sensors and Actuators A: Physical*, vol. 41-42, 1994, pp. 198–206.

[153] A. H. Wright, "Genetic algorithms for real parameter optimization," in *Foundations of Genetic Algorithms*. Morgan Kaufmann, 1991, pp. 205–218.

[154] Y. Xie, G. Loh, B. Black, and K. Bernstein, "Design space exploration for 3D architectures," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 2, no. 2, pp. 65–103, 2006.

[155] N. Yazdi, A. Mason, N. Najafi, and K. Wise, "A low-power generic interface circuit for capacitive sensors," in *Digest of Technical Papers, Solid-State Sensor and Actuator Workshop*, Jun. 1996, pp. 215–218.

[156] ——, "A smart sensing microsystem with a capacitive sensor interface," in *Digest of Technical Papers, IEEE International Symposium on Circuits and Systems*, vol. 4, May 1996, pp. 336–339.

[157] ——, "A generic interface circuit for capacitive sensors in low-power multi-parameter microsystems," in *Sensors and Actuators A: Physical*, vol. 84, 2000, pp. 351–351.

[158] A. Youssif, N. Ismail, and F. Torkey, "Comparison of branch prediction schemes for superscalar processors iceec 2004," *Electrical, Electronic and Computer Engineering, 2004. ICEEC '04. 2004 International Conference on*, pp. 257–260, Sept. 2004.

[159] J. Zhang, K. Zhang, Z. Wang, and A. Mason, "A universal micro-sensor interface chip with network communication bus and highly programmable sensor readout," in *The 2002 45th Midwest Symposium on Circuits and Systems (MWSCAS-2002)*, vol. 2, Aug. 2002, pp. II–246 – II–249.

[160] J. Zhang, J. Zhou, P. Balasundaram, and A. Mason, "A highly programmable sensor network interface with multiple sensor readout circuits," in *Proceedings of IEEE Sensors*, vol. 2, Oct. 2003, pp. 748–752.

[161] C. Zierhofer, I. Hochmair-Desoyer, and E. Hochmair, "Electronic design of a cochlear implant for multichannel high-rate pulsatile stimulation strategies," in *IEEE Transactions on Rehabilitation Engineering*, Mar. 1995, pp. 112–116.

List of Publications

International Conferences

1. C. Strydis, D. Dave, **Identifying Optimal Generic Processors for Biomedical Implants**, *28th IEEE International Conference on Computer Design (ICCD'10)*, Amsterdam, The Netherlands, October 2010, pp. 494-501.

2. C. Strydis, D. Dave, G. N. Gaydadjiev, **ImpBench Revisited: An Extended Characterization of Implant-Processor Benchmarks**, *International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS X)*, Samos, Greece, July 2010, pp. 126-135.

3. D. Dave, C. Strydis, G. N. Gaydadjiev, **ImpEDE: A Multidimensional Design-Space Exploration Framework for Biomedical-Implant Processors**, *21st IEEE International Conference on Application-specific Systems Architectures and Processors (ASAP'10)*, Rennes, France, July 2010, pp. 39-46.

4. C. Strydis, G. N. Gaydadjiev, **Evaluating Various Branch-Prediction Schemes for Biomedical-Implant Processors**, *20th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP'08)*, Boston, MA, USA, July 2009, pp. 169-176.

5. C. Strydis, G. N. Gaydadjiev, **The Case for a Generic Implant Processor**, *30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC'08)*, Vancouver, Canada, August 2008, pp. 3186-3191.

6. C. Strydis, **Suitable Cache Organizations for a Novel Biomedical Implant Processor**, *26th IEEE International Conference on Computer Design (ICCD'08)*, Lake Tahoe, California, USA, October 2008, pp. 591-598.

7. C. Strydis, D. Zhu, G. N. Gaydadjiev, **Profiling of Symmetric-Encryption Algorithms for a Novel Biomedical-Implant Architecture**, *ACM International Conference on Computing Frontiers (CF'08)*, Ischia, Italy, May 2008, pp. 231-240.

8. C. Strydis, G. N. Gaydadjiev, **Profiling of Lossless-Compression Algorithms for a Novel Biomedical-Implant Architecture**, *6th IEEE/ACM/IFIP international conference on Hardware/Software codesign and system synthesis (CODES'08)*, Atlanta, Georgia, USA, October 2008, pp. 109-114.

9. C. Strydis, C. Kachris, G. N. Gaydadjiev, **ImpBench: A Novel Benchmark Suite for Biomedical, Microelectronic Implants**, *IEEE International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (IC-SAMOS VIII)*, Samos, Greece, July 2008, pp. 86-95.

Local Conferences

1. C. Strydis, G. N. Gaydadjiev, S. Vassiliadis, **A New Digital Architecture for Reliable, Ultra-Low-Power Systems**, *ProRISC 2006*, Veldhoven, The Netherlands, November 2006, pp. 350-355.

2. C. Strydis, G. N. Gaydadjiev, S. Vassiliadis, **A Generic Digital Architecture & Compiler for Implantable Devices**, *Architectures and Compilers for Embedded Systems (ACES 2005)*, Ter Elst, Edegem, Belgium, September 2005.

Reports

1. C. Strydis, G.N. Gaydadjiev, S. Vassiliadis, **Implantable Microelectronic Devices: A Comprehensive Review**, *Technical Report (CE-TR-2006-01)*, Delft, The Netherlands, December 2006.

Samenvatting

Gezondheidszorg in de 21e eeuw verandert snel. Met name in ontwikkelde landen, vindt deze verandering plaats van een publieke naar een meer gepersonaliseerde gezondheidszorg. Daarbij nemen de kosten van de gezondheidszorg wereldwijd elk jaar toe. Om betere controle te krijgen over deze kosten, kan en moet beter gebruik van technologie worden ingezet. Op het moment profiteren implantaten al van de verbluffende miniaturisatie technologie, waardoor ze steeds kleiner worden, minder energie verbruiken en de transistors beter kunnen presteren. Deze voordelen brengen echter wel kosten met zich mee. Verder worden in implantaten vaak nog de volgende negatieve verschijnselen geobserveerd: toenemend energie verbruik, afwezigheid van het ontwerp voor betrouwbaarheid en hoge specificiteit waarmee het apparaat kan worden gebruikt. Met het oog op de hiervoor genoegende verschijnselen en in de vooronderstelling dat implantaten een belangrijke rol zullen blijven spelen in de ontwikkeling van een betere, meer gepersonaliseerde gezondheidszorg, zijn wij van mening dat een nieuw paradigma voor het ontwerp van implantaten noodzakelijk is. In dit proefschrift wordt het concept van Smart implantable Medical Systems (SiMS) gedefinieerd. SiMS is een systematische aanpak om biomedische onderzoekers en, hopelijk, de industrie te voorzien van een gereedschapskist met kant-en-klare subonderdelen van implantaten en modellen. Hiermee kunnen implantaten worden gemaakt met een hoge kwaliteit, voor verschillende medische toepassingen. Het SiMS concept moet echter wel aan een aantal essentiële eigenschappen voldoen, namelijk: hoge betrouwbaarheid, modulair ontwerp, zo min mogelijk energieverbruik en miniatuur formaat. Na het SiMS concept te hebben gedefinieerd, zal dit proefschrift de voordelen van de microarchitecturale details van het meest belangrijke SiMS deel onderzoeken: de SiMS processor. In tegen stelling tot de huidige stand van zaken binnen de technologie, zou deze nieuwe processor een universeel en goedkoop zijn, met laag energieverbruik. Verder zou deze processor inzetbaar zijn in een brede verscheidenheid aan medische toepassingen.

Curriculum Vitae

Christos Strydis was born on January 15, 1981 in Athens, Greece. In 1998, he received his high school degree and, by national examinations, was accepted in the Electronics & Computer Engineering (ECE) Bachelor program of the Technical University of Crete, Greece. He majored in Computer Engineering and did his thesis work on Bluetooth-related, embedded systems, for which he was also awarded the 3rd place in the annual "Ericsson Awards of Excellence", Hellas.

In June 2003, he received his 5-year-long diploma with honors and a top-5% ranking in his undergraduate class of 75. In September 2003 he was accepted in the Masters program of the Department of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, with a major in Computer Engineering and a minor in Biomedical Engineering. He became a member of the Computer Engineering (CE) Laboratory chaired by late Prof. S. Vassiliadis and displayed an active interest on biomedical, microelectronic implants, on which topic he also performed his M.Sc. thesis work and graduated his degree with a CGPA of 8.0/10.0 (Honors).

In September 2005, and under the advisory of Assistant Prof. G.N. Gaydadjiev, Christos started his Ph.D. work on processor architectures for microelectronic implants. The research work was funded by the ICT Delft Research Center (DRC-ICT) of the Delft University of Technology and the results of this work are presented in the current dissertation.

Christos is a reviewer for many international conferences and journals, has supervised a number of M.Sc. students to their successful graduation and has served as proceedings co-chair for the SAMOS conference. His research interests include implant-processor design, low-power and dependable computer architectures, and neuroprosthetic applications thereof.